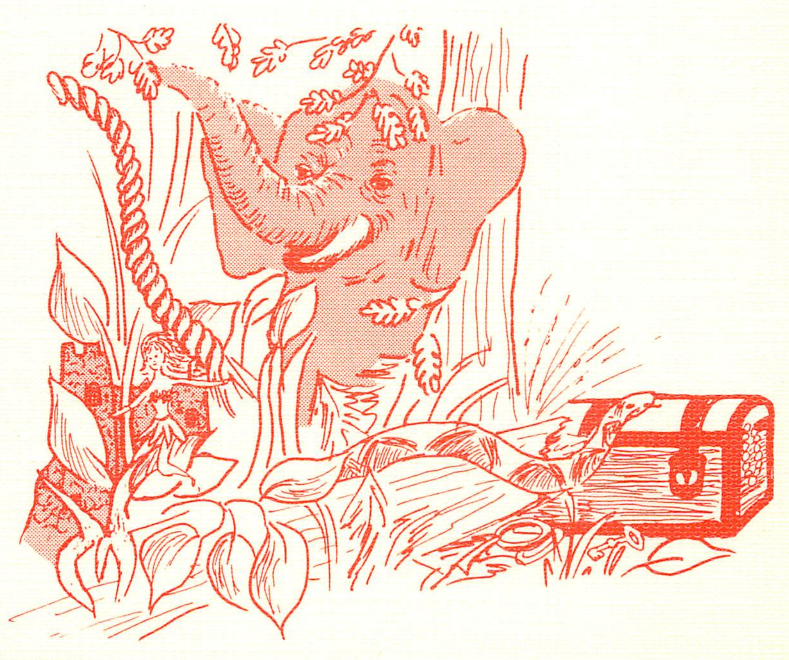
Walkowiak



Adventures

UND WIE MAN SIE PROGRAMMIERT



EIN DATA BECKER BUCH



Walkowiak



UND WIE MAN SIE PROGRAMMIERT



EIN DATA BECKER BUCH

ISBN 3-89011-043-6

Wichtiger Hinweis!

Die in diesem Buch wiedergegebenen Schaltungen, Verfahren und Programme werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden.

Alle Schaltungen, technische Angaben und Programme in diesem Buch wurden von den Autoren mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. DATA BECKER sieht sich deshalb gezwungen, darauf hinzuweisen, daß weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernommen werden kann. Für die Mitteilung eventueller Fehler sind die Autoren jederzeit dankbar.

INHALTSVERZEICHNIS

1. KAPITEL

- EINLEITUNG -

ABENTEUER, sind außergewöhnliche Erlebnisse eines Menschen, die durch extreme Situationen und Gefahren gekennzeichnet sind und sich unauslöschlich der Erinnerung des *Abenteurers* einprägen.
Sie üben einen großen Einfluß auf die Psyche des Menschen aus und faszinieren bereits seit undenklichen Zeiten auch an den Geschehnissen unbeteiligte Personen.
Diese Eigenschaft machte sich die Dichtung zunutze, und der Lauf der Zeit führte zur Entwicklung diverser literarischer Spielarten.
Waren es zunächst Minnesänger, die dem Volke oder auch gekrönten Häuptern von den Taten ferner Helden berichteten, so erwuchsen diese Spielmannsdichtungen im 12. Jahrhundert zu großangelegten Epen wie ´Lanzelet´ oder Wolfram von Eschenbachs ´Parzifal´.
´Don Quijote´ und ´Münchhausen´ begeisterten mit Ihren Erlebnissen ebenfalls Tausende.
Nach dieser Zeit des Schelmenromans machten sich die Reiseerzählungen (Defoe, Cooper, Karl May) das Thema Abenteuer zunutze, bis die Entwiclung mit Kriminal- und Science Fiction Romanen ihren Abschluß fand.

ABENTEUER HEUTE

Wer kennt die eben erwähnten Abenteuerromane nicht, wer hat sich in seiner Kindheit nicht von den Erlebnissen Old Shatterhands oder Robinson Crusoes mitreißen lassen ? - Wer hat sich nicht mit den Hauptpersonen identifiziert und sich vorgestellt, wie man sich wohl selbst in der gleichen Situation verhalten hätte ?

Doch vielleicht gehören Sie tatsächlich zu den wenigen, die aus Gründen der Bequemlichkeit niemals ein solches Buch gelesen haben, warum auch, werden Sie dann fragen, ich kann es doch viel bequemer haben !
Don Quijote habe ich im Fernsehen gesehen, Lederstrumpf und Karl May dank der Programmdirektoren unserer Sender sogar öfter, warum sollte ich also meine Freizeit in irgendeiner Weise aktiv gestalten, wenn es auch anders geht, und außerdem: Lesen ist doch schon längst nicht mehr zeitgemäß.

Sicher, mit dem letzten Argument hätten Sie vermutlich recht, denn waren die Kinderstuben früher mit Puppen, Baukästen, Eisenbahnen oder Kinderbüchern gefüllt, so sind es heute ferngesteuerte Modelle, Experimentierkästen, Videorecorder, Telespiele und Heimcomputer. Unaufhaltsam ist die Technik auch hier auf dem Vormarsch, was bei richtiger Anwendung jedoch durchaus zu begrüßen ist.
Es mag wohl Computerbenutzer geben, die als einzige Anwendung realistische Raumschlachten und diverse andere Schieß- und Actionspiele kennen, doch irgendwann werden auch diese Anwender bemerken, daß der Spielablauf immer der gleiche ist, Denken ist nicht gefragt - nur der Bewegungsrhythmus der Steuertasten ändert sich von Spiel zu Spiel ein wenig.
Also unterzieht man den Softwaremarkt einer neuerlichen Prüfung - und stellt fest, daß die angloamerikanischen Computerbesitzer viel besser bedient werden; denn neben den bekannten Arcade - Games haben sich in Amerika und England längst die Adventurespiele etabliert.

Adventures, zu deutsch Abenteuer, stellen neben Strategiespielen wie Schach oder Othello sicherlich die intelligenteste Anwendung eines Computers zu Spielzwecken dar.
Das Spielfeld besteht nun nicht mehr nur aus einer Handvoll Screens, sondern eine ganze Welt stellt uns nun vor Probleme - vor wirkliche Probleme. Gutes Reaktionsvermögen reicht bei weitem nicht mehr aus, sondern scharfer Verstand und logisches Denkvermögen werden gefordert.
Im Speicher unseres Computers befindet sich nach Laden eines Adventureprogrammes eine eigenständige Welt, eine Welt bestehend aus Räumlichkeiten, Gegenständen und nicht zuletzt aus zahlreichen Mitspielern.
Diese Mitspieler müssen in der Lage sein, sich mit uns verständigen zu können, müssen mit uns einen richtigen Dialog führen können.
Beispielsweise ist da die Hauptperson des Adventures, die Gestalt, mit der wir uns identifizieren sollen. Diese Hauptperson stellen wir uns am besten als eine Art Androiden vor, welchen wir mit Worten unserer Sprache steuern. Wir weisen ihn an, in eine bestimmte Richtung zu gehen, er teilt uns daraufhin mit, was er an seinem neuen Aufenthaltsort sieht. Wir lassen ihn diese Dinge näher untersuchen, befehlen ihm möglicherweise sie zu nehmen, und irgendwann lassen wir ihn diese Gegenstände in geeigneter Form benutzen.

Diese Programme erlauben es uns, ohne große Risiken und Kosten interessante Abenteuer zu erleben, die wie in der realen Welt ablaufen, wir können aber auch in eine phantastische Welt voller Geister und Wunderdinge eintreten; Grenzen sind uns nur durch die Phantasie der Programmierer gesetzt.

Ein gutes Adventure wird uns wie im richtigen Leben agieren lassen, läßt uns sogar den Weltraum, ganz bequem von unserem Lieblingssessel aus, erforschen.

Ist es nicht weitaus interessanter, wenn ich über die Tastatur einer imaginären Person den Befehl *Öffne Tür* erteile und auf dem Bildschirm die Antwort *Ich habe den passenden Schlüssel nicht.* oder *O.K. - Die Tür ist auf* erhalte, als wenn ich nur ununterbrochen den Feuerknopf meines Joysticks drücke und dadurch ein Zählwerk sowie einen Tongenerator in Gang halte ?

Wie interessant es wirklich werden kann, möchte ich Ihnen mit folgenden Beispielen zeigen:

> Wie wollen wir einen Fisch fangen, damit wir nicht verhungern müssen, wenn wir nur unsere bloßen Hände haben?
> Wie können wir den hungrigen Bären davon abhalten, uns als Zwischenmahlzeit zu betrachten?
> Wie soll ich den reißenden Fluß überqueren, der zudem noch von urweltlichen Raubtieren bewohnt wird?

Mit einigem Nachdenken lassen sich all diese Probleme lösen, wir müssen nur die richtigen Ideen und entsprechend viel Phantasie haben:

> Ihre Aufgabe besteht darin, in ein bestimmtes Haus, der Eingang ist durch ein Gittertor gesichert, zu gelangen. Bei genauem Hinsehen sehen Sie einige Zentimeter hinter dem Tor einen Schlüssel liegen, doch leider sind Ihre Arme zu kurz, um durch das Tor hindurch den Schlüssel greifen zu können.
>
> Wie wollen Sie in das Haus hineingelangen ?
>
> Nun, überklettern des Tores ist nicht möglich, aber einige Schritte vor dem Haus waren Ihnen

einige Bäume aufgefallen. Irgendwann im Laufe des Spieles haben Sie außerdem ein Kaugummi gefunden.

Ist der Groschen gefallen ?

Wir haben auch einige Zeit für die Lösung dieses Problemes gebraucht, vor allem, weil uns niemand so deutlich auf verwendbare Gegenstände hingewiesen hatte:

Man brauchte einfach nur zu dem Baum zu gehen und einen Ast abzubrechen. Dann ging man zum Eingang des Hauses zurück, kaute das Kaugummi gut durch, befestigte es an einem Ende des Stockes, schob dieses Ende durch das Gitter des Tores und berührte den Schlüssel, worauf dieser am Kaugummi kleben blieb; durch vorsichtiges Zurückziehen des Stockes gelangte man schließlich und endlich in den Besitz des Schlüssels.
Welch ein Glück, daß dieser dann auch zu dem Tor paßte.

Eine elegante Lösung, nicht ganz einfach, aber vom Programmierer schon während der Entwicklungsarbeit genau so geplant. Ich sollte allerdings ergänzen, daß in der dem Spiel beigefügten Anleitung einige Spielerfahrung vorausgesetzt wurde.

Vielleicht hat obiges Beispiel Sie bereits in ´Abenteuerlaune´ versetzt, sollten Sie die Kraft des geschriebenen Wortes zur Faszination jedoch noch nicht bemerkt haben, dann führen Sie sich bei nächster Gelegenheit doch einmal die *Unendliche Geschichte* zu Gemüte.

Welche Steigerungsmöglichkeiten werden sich nun durch die Implementierung guter Romane in Computersysteme ergeben ?

Vielleicht wird es sich eines Tages als notwendig erweisen, die Erläuterung des Begriffes ´Abenteuer´ in unseren Lexika zu ergänzen:

> Im 20.Jahrhundert ermöglichte die fortgeschrittene Computertechnik den Abenteuerroman zum Mitspielen. Als Meilenstein gilt Scott Adams "Adventureland"; es folgten zahlreiche weitere sogenannte Adventures, die bis zu perfekten Weltsimulationen entwickelt wurden. Zahlreiche Buchautoren mit bekannten Namen (z. B. Michael Crighton) erkannten rechtzeitig die neuen Möglichkeiten und schufen mit ihren Werken für viele Menschen einen Ausgleich zur Alltagswelt.

GESCHICHTE UND ENTWICKLUNG DER ADVENTURES

Nachdem aus der ersten Rechenmaschine ein programmgesteuertes Rechengehirn geworden war, setzten in Programmiererkreisen die unterschiedlichsten Bemühungen ein, den Computern menschliches Denken und menschliche Verhaltensweisen beizubringen. Im Jahre 1966 gelang es schließlich Joseph Weizenbaum vom Massachusetts Institute of Technology mit seinem Programm ELIZA, nicht nur die Fachleute, sondern auch die Öffentlichkeit zu interessieren.
Eliza, in abgemagerten Versionen heute auch für jeden Microcomputer erhältlich, simuliert einen Psychiater und ahmt dessen typische Gesprächstechnik nach. Bei Versuchen mit Testpersonen zeigten sich diese nach der jeweiligen Sitzung sehr überrascht, einer Maschine ihre persönlichsten Probleme mitgeteilt zu haben.

Die weitere Entwicklung der Künstlichen Intelligenz brachte den Fachwissenschaftlern ein auf einer PDP-10 implementiertes Programm: ADVENTURE - Colossal Cave. Zur Abwechslung durften die gelehrten Herren auf einem anderen Gebiet forschen, nämlich nach Schätzen in einer düsteren, von Magie beherrschten Welt. Colossal Cave erfreute sich in Fachkreisen einer recht großen Beliebtheit und wurde schließlich im Sommer 1979 dann auch einer größeren Öffentlichkeit zugänglich gemacht. Die heute wohl jedermann bekannte Firma MICROSOFT veröffentlichte *Microsoft´s Adventure*, eine Diskettenversion dieses Uradventures für den damals in Amerika populärsten Microcomputer, den TRS-80.

Salonfähig gemacht wurden die Adventures jedoch schon im Jahre 1978 von einem jungen Amerikaner namens Scott Adams, welcher mit seinem ADVENTURELAND den Grundstock für den Newcomer ADVENTURE INTERNATIONAL legte.

Auch in diesem Spiel geht es darum, in einer mystischen Welt diverse Schätze zu finden, was natürlich durch Drachen, Labyrinthe, Lavaströme und sonstige Eigen- und Unarten erschwert wird. Dabei fängt alles ganz harmlos mitten im Walde an. Doch wenn man sich erst einmal orientiert hat und weiß, in welche Richtung man sich wenden muß, und dann auch noch den Eingang in das unterirdische Reich findet, dann ...

Die Nachfrage nach ADVENTURELAND war riesig, wohl auch deshalb, weil es noch keine Spiele in Spielhallenqualität gab, und so wurde wegen des großen Erfolges aus diesem ADVENTURELAND rasch eine Reihe von 12 Adventures, wobei durch geschickte Wahl der Thematik das Interesse der Kunden wachgehalten wurde.
Wollten Sie sich nicht auch schon immer einmal mit dem Grafen Dracula anlegen - in *The Count* können Sie es. Oder möchten Sie lieber einen Saboteur in einem Atomkraftwerk finden - kein Problem, *Mission Impossible* macht´s möglich.

Zu dieser Zeit erschien auf dem amerikanischem Softwaremarkt als Mitreiter auf der gleichen Welle ein Programmtyp, der als ´***INTERACTIVE FICTION***´ bezeichnet wurde. Diese Programme sollten einen neuen Literaturtyp darstellen, sollten, wie der Name sagt, Bücher zum Mitwirken sein. Zunächst ist der Spieler nur Leser, er wird nach Programmstart sehr genau in die Handlung eingeführt. Zahlreiche Seiten flimmern über den Bildschirm, die dem Leser Auskunft über Vorgeschichte, augenblickliche Situation, Hintergründe usw. geben, und die sich eben wie ein Buch lesen lassen.

So beginnt *Local Call For Death* wie eine Kriminalgeschichte: Drei Männer treffen sich in ihrem Clubhaus, im Verlauf des Abends wird einer von ihnen ans Telefon gerufen. Es ist der Neffe, welcher den Onkel, sichtlich erregt, um Geld bittet. Der Onkel versagt ihm jedoch die Unterstützung, weiß er doch aus Erfahrung, daß

das Geld auf der Rennbahn verwettet werden wird. Am nächsten Morgen erfährt er vom Selbstmord seines Neffen.

An dieser Stelle übernimmt nun der Spieler. Er stellt ab jetzt eine der Hauptpersonen dar, und versucht im direkten Gespräch mit den Mitwirkenden den vermeintlichen Selbstmord aufzuklären.

Typisch für diese Programme ist die klare Fixierung auf ein einziges Ziel, wie hier die Aufklärung eines Mordes.
Um dieses Ziel zu erreichen, müssen wir als Spieler uns weniger irgendwelcher Gegenstände als geschickter Fragen bedienen. Was ist wohl interessanter, in einem Buch ein Gespräch zu lesen, in einem Film ein Verhör zu verfolgen, oder das Erfolgserlebnis zu haben, einen Verbrecher zu überführen ?

Gerade diese Fähigkeit zu Dialogen macht den Reiz von Interactive Fiction aus, ein Musterbeispiel ist das ebenfalls bei Adventure International erschienene *Encounter in the Park*.
In diesem Spiel begegnen wir während des Morgenspaziergangs unserer Traumfrau, und es wird unser erklärtes Ziel, eben dieses Mädchen zu verführen und dazu zu bringen, uns ihr Ja - Wort zu geben. Der Programmierer hat dem Spieler ein breites Spektrum an Überredungskünsten offengelassen, und es macht wirklich großen Spaß, zu erleben, wie diese Dame auf gewisse Vorschläge reagiert.
Das schlimmste, was uns bei allzu unseriösen Vorschlägen passieren kann, ist, daß sie sich entrüstet von uns abwendet, und wir Mitteilung darüber erhalten, daß wir aus lauter Enttäuschung den Rest unseres Lebens im Kloster verbracht haben.

Wie bereits gesagt, war das Interesse an all diesen Textadventures sehr groß, immer mehr Firmen veröffentlichten demzufolge mehr oder weniger gute Kopien dieser Spiele. Befleißigt durch den zunehmenden Druck der

Konkurrenz, ließen sich daher einige Softwarehersteller neue Generationen von Adventures einfallen.
Es erschienen die ersten ***LABYRINTH - ADVENTURES*** mit dem Spielziel, aus einem Gebäude zu entkommen, jedoch verirrte der Spieler sich fast immer hoffnungslos in dem perspektivisch auf dem Bildschirm dargestelltem Labyrinth.
Nächste Stufe der Entwicklung waren die ***QUEST - ADVENTURES***, welche dann meistens auch gleich als Real - Time Adventures implementiert wurden. Dabei wird der Spieler einem zusätzlichen Streßeffekt unterworfen, denn wenn er beim Erscheinen irgendwelcher Räuber nicht sofort F für Fight eingibt und danach nicht laufend irgendwelche Tasten oder gar Tastenkombinationen drückt, um sein Schwert zu schwingen, ist das Spiel bereits sehr frühzeitig wegen zu starker Beschädigung der Hauptperson beendet.
Aus der Konzeption dieser Spiele ergibt sich leider eine starke Einschränkung des Handlungsspielraumes des Spielers, denn es bleiben ihm nur einige wenige Aktionen wie Kämpfen, Verhandeln, Kaufen oder Flüchten, welche ihm per Menü, daher QUESTion, angeboten werden.

Die nächste Entwicklungsstufe der Adventurespiele wurde dank des Preisverfalls für Grafikprozessoren und RAM - Bausteine möglich, denn nun war es für die Computerhersteller nicht mehr schwierig, dem Heimanwender bei preisgünstigen Computern eine grafische Leistung zur Verfügung zu stellen, die früher nur auf vielfach größeren und somit auch teureren Anlagen möglich war.

So begann im Jahre 1982 (für Apple - Benutzer etwas früher) mit dem Vertrieb des ersten ***GRAFIKADVENTURES*** ein neuer Aufschwung für diese Spiele. Denn waren sie von vielen Computerbesitzern als reine Textspiele bislang verpönt worden, so waren sie nun wegen der vielen bunten Bilder auf einmal interessant; darüber hinaus erwiesen sie sich als hervorragend dazu geeignet, der computerunkundigen Verwandschaft auf die Frage *´Was kann denn dein Computer ?´* zumindest ein erstauntes *´Oh, ist ja schön !´* zu entlocken.

An Spielwert und Ziel hatte sich natürlich überhaupt nichts geändert. Plötzlich waren also weitere Käuferschichten erreichbar; ein Grund, viele jahrelang bekannte Adventures im neuen Gewande auf den Markt zu werfen: die ausführlichen Beschreibungen waren verschwunden, statt dessen war die augenblickliche Lage auf dem Fernsehschirm zu sehen.

Ob die Programme dadurch besser oder einfacher spielbar geworden waren sei dahingestellt, ich persönlich ziehe es jedoch vor, durch eine Mitteilung wie *Ich bin im Wald. Um mich herum stehen lauter Bäume. Zwischen zwei Eichen sehe ich ein Erdloch.* auf wichtige Dinge hingewiesen zu werden, als im Unterschied dazu nur einige Bäume zu sehen; diese halte ich dann verfrüht für ein ganz normales Stück Wald, weshalb ich auch schnell weitergehe, die Bäume nicht weiter untersuche und dadurch das Erdloch, einen kleinen schwarzen Flecken auf dem Monitor, natürlich übersehe. Selbstverständlich liegt ausgerechnet darin ein Teil des Schatzes oder aber ein Gegenstand, ohne den eine erfolgreiche Beendigung des Spieles nicht möglich ist.

2. KAPITEL
- DAS KONZEPT -

DIE AUFMACHUNG DER ADVENTURES

In den folgenden Kapiteln dieses Buches werden wir gemeinsam Adventurespiele entwickeln. Selbstverständlich wünschen wir für unser Werk ein möglichst professionelles Aussehen, daher werden wir nicht darauf verzichten können, Eigenschaften und spezielle Features auf dem Markt befindlicher Programme zu analysieren.
Anschließend werden wir uns dann Gedanken darüber machen, wie wir die einzelnen Funktionen in BASIC programmieren und gegebenenfalls sogar erweitern können.

Es wurde bereits herausgestellt, daß ein Adventure den Spieler in eine andere Welt versetzen will, in eine Welt, die das Unmögliche möglich macht. Er soll hier Abenteuer erleben, muß sich also bewegen und handeln können, ebenso müssen ihm die Resultate seiner Bemühungen mitgeteilt werden. Sinnvolles und zielgerichtetes Handeln ist dem Menschen jedoch erst nach Gebrauch seiner Sinne möglich. Die Informationen, die der Abenteurer im wirklichen Leben mit seinen Augen, Ohren und seinem Tastsinn aufnimmt, müssen dem Spieler daher in geeigneter Form angeboten werden.
Da das menschliche Gehirn bis heute nicht direkt an einen Computer angeschlossen werden kann, erhalten wir Antworten auf die Fragen *Wo bin ich ? Was sehe ich ? Wohin kann ich gehen ? Was fühle ich ?* und *Was sehe ich ?* daher in Gestalt kurzer, aber vollständiger Sätze.

Die Erfassung der Umwelt wird allerdings in den seltensten Fällen bewußt ablaufen, daher kann auch dem Adventurespieler nicht zugemutet werden, umfangreiche Abhandlungen über den gerade betretenen Raum zu studieren, sonst wird er sich nie mit der Spielfigur identifizieren und sich der Illusion einer Scheinwelt hingeben.
Also wird das Monitorbild, sofern es sich nicht um Grafikspiele handelt, in zwei übersichtliche Zonen

aufgeteilt, von denen jede eine genau definierte Funktion hat.

So wird uns die obere Bildhälfte darüber Auskunft geben, wo wir uns gerade befinden und was wir sehen, die untere Hälfte ist hingegen für die Kommunikation mit der für uns handelnden Hauptperson des Adventures bestimmt. Hier geben wir unsere Anweisungen ein, und hier erfahren wir, was auf unsere Eingaben hin geschieht.

Auf diese Art und Weise wird uns in einer imaginären Welt die gezielte Fortbewegung ermöglicht, wie uns auch die Möglichkeit zu sinnvollen Manipulationen und Handlungen gegeben wird.

Ein typisches Schirmbild könnte etwa so aussehen:

Ich bin in einem düsteren Wald.
Ich sehe viele alte Bäume.

Ich kann nach Norden, Osten, Westen.

Was soll ich tun ?

Für den Fall, daß wir beabsichtigen, in dem Spiel möglichst rasch weiterzukommen, wäre es falsch, ziellos in eine der drei Himmelsrichtungen weiterzumarschieren. Das Nächstliegendste, was ein Adventurespieler in jeder

Situation tun sollte, ist, alles auf das genaueste zu untersuchen. So wären mögliche Antworten auf *Untersuche Baum* etwa: *Es handelt sich um alte Eichen.* oder *Auch hier greift das Baumsterben um sich.*
Diese Antworten helfen uns zwar nicht weiter, zeigen sie uns doch scheinbar nur, daß es doch nicht immer so wichtig ist, alles zu untersuchen, doch woher soll man die Antwort vorher wissen ? Denn auch eine Antwort wie *In einem Stamm sehe ich eine Öffnung.* wäre möglich gewesen. Vielleicht handelt es sich um einen alten Spechtbau, in welchem sich ein Teil des gesuchten Schatzes befindet; *Untersuche Öffnung* wird es an den Tag bringen.

Hätten wir keine Anhaltspunkte gefunden, auch auf *Untersuche Wald* hin - es könnte ja ein Zauberwald sein - nicht, müßten wir uns entscheiden, wohin wir uns nun sinnvollerweise begeben wollen.
Wie würden Sie sich in solch einer Situation verhalten, ohne Karte und Kompaß, hilflos und allein, weitab aller Wege? - Vermutlich würden Sie auf einen Baum klettern, um zu sehen, was sich in der näheren und weiteren Umgebung befindet.

Ihre Eingabe sollte also lauten: *Besteige Baum*, und wenn Sie nicht eine Mitteilung in der Art *Dazu bin ich nicht sportlich genug !* erhalten, wird sich das Bild auf dem Monitor ändern:

Ich bin in der Krone einer alten Eiche.
Ich sehe im Norden ein Gebirge,
im Osten einen See,
im Westen steigt Rauch empor.

Ich kann nach unten.

BESTEIGE BAUM

Okay !

Was soll ich tun ?

Damit sieht die ganze Angelegenheit doch schon viel besser aus, denn nach Eingabe von *U* für Unten haben wir nur noch das Problem, zu entscheiden, welche Gegend wir zuerst besuchen wollen.

Die Mitteilung *Okay !* informiert uns darüber, daß die Eingabe verstanden und ausgeführt worden ist. Wäre dem nicht so, hätten wir beispielsweise *Ich verstehe das Verb nicht !* erhalten, womit uns das Programm sicherlich zu einer anderen Ausdrucksweise veranlassen würde. Gleiches gilt selbstverständlich für das Objekt unserer Eingabe.

Denkbar sind allerdings auch noch zwei weitere Antworten. Zum einen werden wir während eines Spieles oft die Mitteilung *I must be stupid, but... - Ich verstehe nicht, was Du meinst !* erhalten, womit das Programm uns mitteilen will, daß die benutzten Worte zwar zum programmierten Wortschatz gehören, der Befehl im Moment jedoch nicht sinnvoll ist: *Nimm Schlüssel* wird vom Adventureprogramm verstanden werden, wenn ein Schlüssel im Adventure auftritt, wird jedoch sinnlos, falls sich dieser Schlüssel nicht im gleichen Raum wie der Spieler befindet.
Ähnlich verhät es sich mit der zweiten Standardmeldung *You can't do that ... yet. - Das geht im Moment nicht.* Sollten wir diese Antwort erhalten, dürfen wir aufatmen, denn wir sind auf dem richtigen Weg; es sind nur noch nicht alle erforderlichen Bedingungen erfüllt, um den Befehl auszuführen. So wird sich keine verschlossene Tür öffnen lassen, solange wir den zugehörigen Schlüssel nicht haben.

STANDARDFUNKTIONEN DER ADVENTURES

In einem durchschnittlichen Adventure finden wir 30 - 50 verschiedene Räumlichkeiten, in jedem Raum einen oder mehrere Gegenstände. Diese Gegenstände können wiederum irgendwelche Sachen enthalten, und da man im Adventure nie genau weiß, was ein paar Züge weiter unbedingt gebraucht wird, neigen viele Adventurespieler dazu, alles mitzunehmen, was ihnen zwischen die Finger kommt und nicht zu schwer zum Tragen ist.
Den Überblick zu behalten, wird dabei recht schwierig, denn da wir alle bequem geworden sind, machen wir uns natürlich keine Notizen.
Diese Erkenntnis kam den Programmierern glücklicherweise recht früh, so daß heute jedes Adventure nach Eingabe von *INVENTUR* alle Gegenstände, die wir mit uns führen, auf dem Bildschirm ausdruckt.

Falls es sich um ein freundliches oder speziell für Anfänger geschriebenes Adventure handelt, können wir uns in verfahrenen Situationen Tips geben lassen, denn dann ist der Befehl *HILFE* implementiert.

Im Gegensatz zur Standardeingabe aus Verb und Objekt handelt es sich hier wie bei Inventur um einen sogenannten Ein - Wort Befehl, welcher darüber hinaus nicht einmal eine Aktion einleitet, doch was kann dieses eine Wort nicht alles bewirken:

Ich würde alles untersuchen zeigt uns unsere Nachlässigkeit, die Mitteilung *Eine feenhafte Gestalt erscheint und schreibt den Satz ´Sesam öffne dich´ in den*

Sand wird uns hingegen in einer verfahrenen Situation zu einem Freudensprung veranlassen.
Schlechter sieht es aus, wenn wir für die Hilfe bezahlen sollen: *Ein Troll erscheint und sagt, daß uns für seine Hilfe 10 Punkte abgezogen werden. Er will wissen, ob wir Hilfe benötigen ?*
Dunkle Wolken werden vermutlich in den Fällen am Horizont auftauchen, wenn wir auf die Eingabe von Hilfe folgende Nachricht erhalten: *'Hilfe' ist in diesem Adventure nicht vorgesehen !* oder wenn als Antwort nur die allgemeine Spielanweisung wiederholt wird.

Sicherlich mag es ärgerlich sein, wenn man an einer gewissen Stelle trotz aller Mühen nicht weiterkommt, jedoch sollte niemand die letzten zwei Beispiele als einen Akt der Boshaftigkeit des Programmierers ansehen; wäre die Hilfsfunktion durchgehend gewährleistet, würden einige ganz Schlaue das Adventure in 30 Minuten lösen, dabei aber niemals die Faszination eines solchen Programmes kennenlernen.

VORÜBERLEGUNGEN ZUR REALISATION

Der generelle Aufbau der Adventurespiele ist uns nun bekannt. Wir haben eine Vorstellung davon, wie unser Adventure sich auf dem Monitor zu präsentieren hat, und welche Unterstützungsfunktionen wir standardmäßig implementieren müssen. Im folgenden wollen wir uns daher Gedanken zu den einzelnen Elementen unseres Adventuresystems und zu deren programmtechnischer Verwirklichung machen.

RÄUME im Adventure:

Die Adventurewelt setzt sich aus den Räumen des jeweiligen Spieles zusammen, d.h. jeder durch den im Adventure herumwandernden Spieler erreichbare Ort wird als Raum bezeichnet. Dabei ist dessen Größe völlig unbedeutend, es kann, wie in unseren bisherigen Beispielen, ein Wald, bzw. eine Baumkrone gemeint sein, es kann sich jedoch auch um das Innere eines Autos oder Kleiderschranks handeln.
Die einzelnen Räume unterscheiden sich für den Spieler durch ihre Beschreibungen und ihre geografische Anordnung; programmtechnisch unterscheiden sie sich, wie wir noch sehen werden, durch ihre Raumnummer.

Alle diese Räume sind für den Spieler erreichbar, müssen also in geeigneter Weise miteinander verbunden werden. In jedem Adventure ist die Bewegung in die vier Himmelsrichtungen Norden, Süden, Osten und Westen möglich, einige erlauben zusätzlich Bewegungen in der Vertikalen.
Selbstverständlich darf es jedoch nicht geschehen, daß der Spieler Raum 5 in westlicher Richtung verläßt, Raum 6 dabei von Osten her betritt, und wenn er noch einmal in Raum 5 will, keinen Weg in Richtung Osten findet, oder, schlimmer noch, wenn er durch eine Bewegung nach unten wieder in Raum 5 gelangt.

Ausnahmen bestätigen auch hier die Regel, denn was wäre ein Adventure ohne ein magisches Labyrinth.

OBJEKTE

Typisch für jeden Raum sind weiterhin die in ihm befindlichen Objekte. Wir müssen alle Gegenstände in den Räumen unterbringen, oder, korrekter ausgedrückt, jedem Objekt einen Raum zuordnen. Bei dieser Zuordnung vermeiden wir automatisch, daß irgendwelche Objekte doppelt existieren.

Ein weiteres Problem stellt sich uns mit der Unterbringung derjenigen Objekte, die bei Spielstart gar nicht im Adventure vorhanden sein dürfen. - Wo bleiben sie bis zu ihrem Auftreten ?

Angenommen, unser Held betritt einen Raum, und als sichtbares Objekt wird nur eine alte, verschlossene Holzkiste ausgegeben.
Auf die Eingabe des Spielers *Öffne Kiste* verlangen sämtliche Regeln der Logik, daß die alte, verschlossene Kiste vom Bildschirm verschwindet, dafür sollte jedoch eine alte, geöffnete Kiste auftauchen, die womöglich mit Silbermünzen gefüllt ist.

Nun, die Lösung dieser Probleme wird uns keine allzu großen Schwierigkeiten machen. Wir werden in unserem Adventure einen zusätzlichen Raum, den **LAGERRAUM**, installieren und in diesem alle zur Zeit nicht benutzten Objekte lagern. Wenn wir keine Verbindungen zu diesem Raum hin programmieren, kann der Spieler ihn auch nie betreten.

Im nächsten Kapitel werden wir sehen, wie einfach der Lagerraum und auch die gesamte Adventurewelt zu programmieren ist; weiterhin werden wir im Verlauf des

Buches die Notwendigkeit weiterer ***BESONDERER RÄUME*** einsehen.

EINGABEN

Wie bereits aus dem voranstehenden Text hervorgegangen ist, bestehen die Eingaben des Spielers meist aus einem Verb und einem Objekt. Unser Programm muß daher zunächst testen, ob die Eingabe erlaubt ist, wozu eine Trennung in Verb und Objekt erforderlich wird. Sollte der Spieler sich verschrieben haben, oder das benutzte Verb nicht vorgesehen sein, muß eine entsprechende Meldung gemacht werden. Findet unser Adventure das Eingabeverb innerhalb seines Wortschatzes wieder, muß das Objekt auf gleiche Weise überprüft werden.
Sind beide Worte definiert, kann das Programm in vorgeschriebener Weise reagieren.

Bei der Erstellung dieses Wortschatzes muß der Programmierer übrigens besondere Sorgfalt walten lassen, er muß alle möglichen Eingaben, die irgendein Spieler machen könnte, vorhersehen. Die Einbringung von Synonymen wird erforderlich, denn sollte der Spieler allzu lange damit beschäftigt sein, dem Adventure seine Wünsche klarzumachen, wird er schnell das Interesse verlieren.

Gerade für uns Deutsche kann die Wahl der Worte jedoch zum Problem werden, denn wo der Amerikaner sich mit *Climb Tree, Shoot Gun, Drop Box, Look* leicht verständlich machen kann, sieht es für uns nicht ganz so eindeutig aus: *Kletter Baum, Schieß Gewehr, Leg Kiste, Sieh.*

MITTEILUNGEN

Die Ausführung einer Reaktion muß dem Spieler natürlich deutlich gemacht werden. Im einfachsten Fall geschieht dies durch O.K., meist werden jedoch zusätzliche Mitteilungen ausgegeben. Dabei werden die unterschiedlichsten Eingaben oft die gleiche Antwort erforden, weshalb es sinnvoll ist, eine Tabelle mit bestimmten Standardmeldungen wie: *So stark bin ich nicht.* oder *Das hat doch wohl keinen Sinn!* anzulegen.

DIE SPIELIDEE

Damit haben wir das Mindestmaß an grauer Theorie bewältigt, jetzt können wir unseren Commodore 64 einschalten und mit der Praxis beginnen. Vielleicht haben Sie bereits eine feste Vorstellung von Ihrem ersten Adventure, wenn nicht, lassen Sie sich doch durch folgende Vorschläge inspirieren.

1. DAS SPUKHAUS

Sie erhalten den letzten Brief von Ihrer Tante, ein Schreiben, in dem sie Ihnen ihr altes hochherrschaftliches Haus vererbt. Stutzig macht Sie jedoch der Hinweis, daß Sie sich vorsichtig verhalten sollen, denn sonst könne es Ihnen wie Ihrem Onkel ergehen. Sie lassen sich durch solcherlei Gerede nicht davon abhalten, das Haus aufzusuchen. Sogleich stellen Sie allerlei Ungereimtheiten fest, finden eine geheime Bibliothek mit Büchern über Geisterbeschwörung, entdecken im Keller eine Opferstätte. Zahllose Geister machen Ihnen das Leben schwer, bis Sie schließlich entdecken, daß einer Ihrer Vorfahren den Sektenführer zum Ausbau der eigenen Macht ermordet hatte und zur Strafe lebendig eingemauert worden war. Sie sollen ihm nun zur ewigen Ruhe verhelfen.

2. DAS VERMÄCHTNIS DES ALTEN

Ihr Freund und Nachbar, ein berühmter Wissenschaftler, ist unter mysteriösen Umständen ums Leben gekommen. Sie erhalten eine Notiz, in der er Sie bittet, sein Lebenswerk zu vollenden. Nachdem Sie es geschafft haben, sein bislang geheimes Labor zu betreten, ist es Ihre Aufgabe, seinen Supercomputer in Betrieb zu nehmen. Dieser gibt Ihnen dann weitere Hinweise.

4. *STAR ODYSSEY*

Ihr Raumschiff wird durch einen kosmischen Sturm so stark beschädigt, daß Sie auf einer unbemannten Station landen müssen, um Ersatzteile zu beschaffen, damit Sie zur Erde zurückkehren können.

4. *GOLDRAUSCH*

Sie treffen in der Wildnis auf einen tödlich verletzten alten Mann. Dieser macht Ihnen Mitteilung über seine Goldmine. Er zeigt Ihnen einige Goldstücke und berichtet Ihnen, daß er an sieben verschiedenen Stellen auf dem Minengelände weiteres Gold versteckt hätte; da er es nicht mehr benötige, könnten Sie es sich holen.

Diesen zuletzt gemachten Vorschlag, ein Adventure des Typs Schatzsuche, möchte ich als Grundlage für die nächsten Kapitel dieses Buches nehmen, und gemeinsam mit Ihnen ausarbeiten. Selbstverständlich steht es Ihnen frei, eine eigene Idee zu realisieren, jedoch werde ich bei allen noch vorzustellenden Erweiterungen immer wieder Bezug auf *Goldrausch* nehmen.

3. KAPITEL

- DIE VERWIRKLICHUNG -

DIE GESTALTUNG DER WELT

Mit der Wahl des Themas steht der grobe Ablauf des Adventures bereits fest, im folgenden werden wir daher einzelne Strukturen ausarbeiten und diese wiederum immer weiter verfeinern, bis eine bis ins kleinste geplante Welt geschaffen worden ist.

Genauso wollen wir auch bei der Programmierung verfahren; Schritt für Schritt werden wir unser Adventure aufbauen und uns immer wieder von der korrekten Funktion der einzelnen Routinen überzeugen.

Um jedoch gegebenenfalls einzelne Programmteile austauschen oder ergänzen zu können, möchte ich Sie an dieser Stelle bitten, nur die vorgegebenen Zeilennummern zu verwenden.

DIE KARTE

Am Anfang steht zunächst das Nichts. Doch werden wir die für unser Spiel so dringend benötigte Welt im folgenden nach unseren Vorstellungen schaffen. Was unsere Welt bieten muß, legt das Thema fest. Goldrausch wird vermutlich in und um einem Bergwerk stattfinden.

Stellen wir uns die Mine einmal vor: ein zerklüfteter Abhang voller Geröll; ein Werkzeugschuppen; Holzrinnen, die das für die Goldwaschung benötigte Wasser eines Gebirgsbaches heranführen; morsche, trockene Holzbalken, die den düsteren Eingang säumen; dunkle Gänge voller Gefahren.

Sicherlich werden uns ohne große Schwierigkeiten noch zahlreiche ähnliche Bilder einfallen, die uns eine für die

vorgesehene Handlung ausreichende Menge Räume finden lassen werden.

Unser konkretes Beispiel möchte ich zunächst auf nur sechs Räume beschränken. Beginnen wollen wir mit einer Wanderung durch einen Wald bis wir zur Mine gelangen, wo unsere Aufgabe darin bestehen wird, die Schätze zu suchen.

Damit ist der vorläufige Beginn von Goldrausch festgelegt. Um jedoch über Sieg oder Niederlage des Spielers entscheiden zu können, müssen wir ebenfalls ein erfolgreiches Spielende festlegen, mit weiterem Fortschreiten unseres Projektes natürlich auch mehrere verlustbringende Situationen.

So können wir uns mit dem Finden aller Schätze begnügen, können dem Abenteurer aber auch die Aufgabe stellen, den gefundenen Schatz an einem sicheren Ort zu deponieren. Zunächst soll das Auffinden des Goldes jedoch ausreichend sein.

Aufgabe und Ziel stehen fest, ebenso zwei wesentliche Handlungsorte. Da ist zum einen der Wald, den wir aus zwei Räumen aufbauen werden, und zum anderen die Mine. Allerdings ist uns klar, daß wir in dieser kleinen Welt keine Handlung ablaufen lassen können. Denn wir brauchen Platz, um einige Gegenstände verstecken und dem Spieler einige Fallen stellen zu können.

Ein Bergwerk wird normalerweise nicht zwischen den Wurzeln eines Baumes beginnen, wir müssen geeignete Übergänge der Landschaft schaffen, um ein realistisches Spiel zu ermöglichen.

Drei weitere Räume stellen uns zur Ausarbeitung der ersten Version unseres Adventures genügend Platz zur Verfügung, wir können nun eine Liste aller auftretenden Orte formulieren:

im Wald
im Wald
durch einen Felshang begrenzter Waldrand
eine Waldlichtung
Waldlichtung am Berghang
Eingang ins Bergwerk

Der nächste Schritt zum fertigen Programm ist zweckmäßigerweise die Anfertigung einer Karte, in der alle Orte als Rechtecke dargestellt werden. Diese Rechtecke werden durchnumeriert und mit ihrer Raumbeschreibung versehen, wobei wir bereits auf eine passende Formulierung achten und auch nach Möglichkeit die Zeilenlänge von 40 Zeichen unseres Commodore 64 berücksichtigen. So muß sich unsere Beschreibung reibungslos an den einleitenden Text *Ich bin* anfügen.

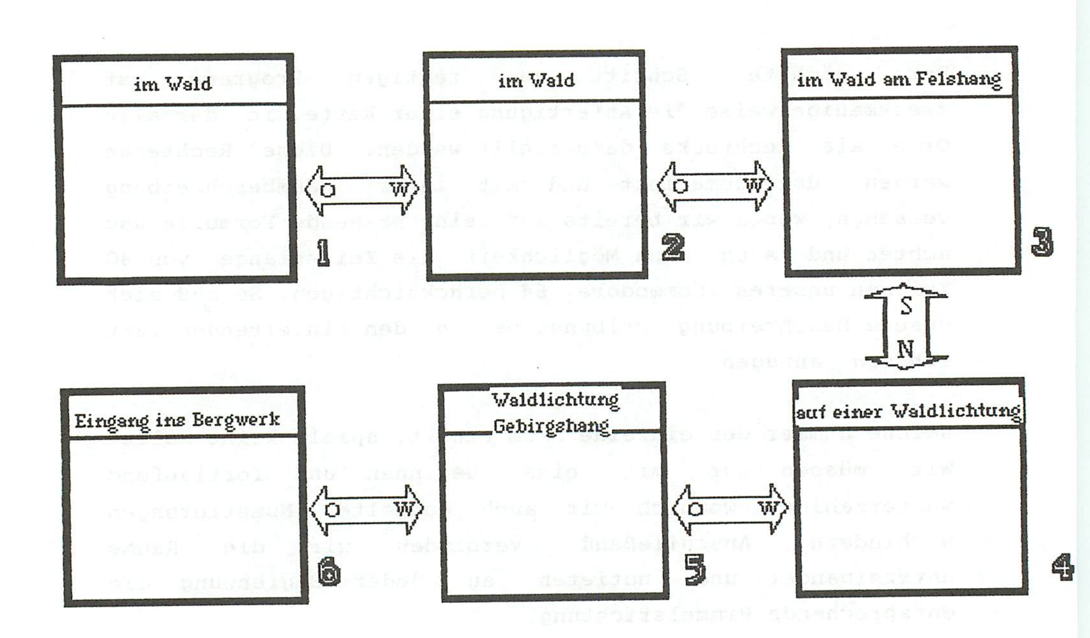
Welche Nummer der einzelne Raum erhält, spielt keine Rolle. Wir müssen nur mit eins beginnen und fortlaufend weiterzählen, wodurch wir auch doppelte Numerierungen verhindern. Anschließend verbinden wir die Räume untereinander und notieren an jeder Begrenzung die entsprechende Himmelsrichtung.

Aus dieser Karte läßt sich nun ohne weiteres ablesen, daß, wenn der Spieler sich beispielsweise in Raum 3 befindet, folgende Mitteilungen auf dem Monitor ausgegeben werden müssen:

Ich bin im Wald. Vor mir steigt ein steiler Felshang auf.

Weiterhin gibt uns unsere Karte Aufschluß über die möglichen Bewegungsrichtungen:

Ich kann nach Westen, Süden.


im Wald
im Wald
im Wald am Felshang
O
W
O
W
1
2
3
S
N
Eingang ins Bergwerk
Waldlichtung
Gebirgshang
auf einer Waldlichtung
O
W
O
W
6
5
4

In unserem Programm könnten wir nun für jeden Raum eine Bedingung der Art *Wenn Spieler in Raum 3, dann drucke 'Ich bin im Wald. Vor mir steigt ein steiler Felshang auf'* programmieren, würden damit unseren Speicher aber schon nach kurzer Zeit gefüllt haben.

Das Prinzip ist jedoch richtig, Wir werden die Raumbeschreibungen in einem Variablenfeld, dem wir sinnvollerweise den Namen RAUM$ geben, abspeichern.

EXKURS: VARIABLEN & FELDER

Um Zwischenergebnisse oder Daten eines Programmes zu speichern, werden in jeder Computersprache Variablen verwendet.
Diese Variablen verhalten sich wie kleine Fächer, in denen etwas zur Aufbewahrung abgelegt werden kann. Diese Fächer haben Namen, Kofferraum oder Schreibtischschublade wären möglich. Dabei gibt es dann Fächer gleicher Art: am Schreibtisch untereinander Schublade 1, Schublade 2, 3, usw. Jedes Fach heißt Schublade und unterscheidet sich vom nächsten nur durch seine Nummer. So auch in Basic: Wir können jeder Variablen einen eigenen Namen geben, oder auch gleichartige mit einem einzigen Namen und einer Indexnummer versehen, diese wird dann in Klammern hinter die Bezeichnung der Variablen geschrieben.
Dabei lassen sich zwei grundsätzliche Variablentypen unterscheiden. Die einen nehmen nur Zahlen auf, Zahlen, mit denen auch gerechnet werden kann; die anderen Variablen dienen der Aufnahme von Texten. Zusammensetzung und Länge der Texte spielen keine Rolle, genauer: wir wollen jedes Zeichen auf unserer Tastatur und Längen bis zu 255 Zeichen hintereinander erlauben.
Einige Beispiele für korrekte Variablen wären: A, A1, A2, TEXT1 (nur für Zahlen), TEXT1$ (für Texte) als einfache Variablen, A(1), A(2), A(3) als indizierte Variablen.

Vielleicht haben Sie auch schon einige unserer Mitmenschen gesehen, welche versuchen, ihre Wohnung durch das Aufstellen von mit kleinen Miniaturen gefüllter Setzkästen - sie wurden für die Aufbewahrung der kleinen Lettern in einer Druckerei geschaffen - zu verschönern; wir Anfänger stellen uns jetzt einmal jedes einzelne Kästchen als eine Variable vor.
Um gleich ein praktisches Beispiel zur Hand zu haben, führen wir, besser nur in Gedanken, folgendes durch:
Aus unserer Adventurekarte schneiden wir die Rechtecke der einzelnen Räume aus. Diese legen wir in jeweils einen Kasten, achten dabei jedoch darauf, daß die frühere Anordnung der Räume erhalten bleibt, d.h. ganz oben links liegt unser Wald, in der zweiten Reihe links außen liegt der Eingangsstollen.
Auf die Frage, wo das Bergwerk sei, könnten wir mit ´in der ersten Spalte der zweiten Reihe´ antworten, ebenso sind die Positionen aller anderen Räume durch die Koordinaten Reihe und Spalte festgelegt. Diese Anordnung wird in Computersprachen als zweidimensionales Feld bezeichnet, oder, viel schöner, als Array.
In Basic würden wir unseren Setzkasten als Variablenfeld KARTE$ ($ weil Texte gespeichert werden sollen) bezeichnen, und den einzelnen Variablen folgenden Inhalt zuordnen:
KARTE$(1,1) = "im Wald" ´erste Reihe u. erste Spalte
KARTE$(1,3) = "Waldlichtung" erste Reihe, dritte Spalte
KARTE$(2,2) = "Eingang Bergwerk" usw.

Ab Zeile 100 geben wir daher folgenden Programmtext ein:

```
100 RAUM$(1) = "im Wald."
110 RAUM$(2) = "im Wald."
120 RAUM$(3) = "im Wald vor einem Fels
hang."
130 RAUM$(4) = "auf einer Lichtung."
140 RAUM$(5) = "auf einer Lichtung am Ra
nde eines Berghanges."
```

```
150 RAUM$(6) = "am Eingangsstollen eines
 alten Bergwerkes."
```

Damit existieren die Räume in unserem Computer, es fehlt uns noch die Spielfigur, welche sich in der programmierten Welt bewegt und uns die Situation, in der sie sich gerade befindet, beschreibt. Ihr Aufenthaltsort wird natürlich von unseren Eingaben abhängig sein; wir führen daher eine weitere Variable, die wir SPIELER nennen wollen, ein, diese speichert die jeweilige Position des Spielers.

Sinnvolle Werte für diese Variable sind bislang die Zahlen eins bis sechs, denn PRINT RAUM$(SPIELER) druckt dann die jeweilige Beschreibung des Aufenthaltsraumes aus. Geben wir zu Testzwecken die nächsten Zeilen ein:

Achtung ! Ich möchte Sie noch einmal bitten, die angegebenen Zeilennummern unbedingt beizubehalten, da einige der Zeilen mit der weitereren Entwicklung verändert werden, sowie nicht benutzte Nummern durch zusätzliche Programmzeilen belegt werden!

```
1140  PRINT "Ich bin ";
1150  PRINT RAUM$(SPIELER)
1280 INPUT "SPIELER IN WELCHEN RAUM";SPI
ELER
5000  GOTO 1140
```

Erkundungen unserer Welt werden dadurch möglich, doch ist die Art der Weiterbewegung keineswegs befriedigend. Wir wollen nicht wahllos von einem Raum in einen anderen springen, sondern streben eine gezielte Orientierung nach den Himmelsrichtungen an.

Normalerweise wird der gerade vom Spieler besuchte Raum mindestens einen, maximal sechs Ausgänge haben. An Raum 1 grenzt im Osten Raum 2 an, in die Richtungen Norden, Süden und Westen, ebenso nach oben und unten, führt kein Weg. Wenn wir darin übereinkommen, daß die Himmelsrichtungen stets in der Reihenfolge N, S, W, O, Oben und Unten genannt werden, kann Raum 1 folgendermaßen spezifiziert werden:

1 im Wald -,-,-,X,-,-

Zweckmäßigerweise werden Ausgänge dann nicht durch ein X markiert, sondern es wird explizit der in dieser Richtung zu erreichende Raum aufgeführt:

1 im Wald 0,0,0,2,0,0
2 im Wald 0,0,1,3,0,0

Diese sechs Werte werden für jeden Raum in einer Variablen, nennen wir sie DURCHGANG, gespeichert. Da hier zwei verschiedene Werte (Raum und Richtung) den zu betretenden Raum kennzeichnen, ist die Speicherung in einem zweidimensionalem Feld angebracht, wobei die Reihe jeweils gleich dem Raum ist und die Spalten eins bis sechs den möglichen Wegen entsprechen.

RAUM	N	S	W	O	OB	U
1	0	0	0	2	0	0
2	0	0	1	3	0	0
3	0	4	2	0	0	0
4	3	0	5	0	0	0
5	0	0	6	4	0	0
6	0	0	0	5	0	0

Es dürfte niemandem schwerfallen, diese Richtungstabelle (auch Travel - Table genannt) zu programmieren.

Anfänger, welche die nächsten Zeilen nicht verstehen, sehen sich bitte unser vorangegangenes Setzkastenbeispiel noch einmal an.

```
101 DURCHGANG(1,1)=0 : DURCHGANG(1,2)=0
102 DURCHGANG(1,3)=0 : DURCHGANG(1,4)=2
103 DURCHGANG(1,5)=0 : DURCHGANG(1,6)=0
```

Obige Zeilen programmieren den Durchgang von Raum 1 nach Raum 2 in östlicher Richtung. Entsprechend könnten wir mit den anderen Räumen verfahren, würden dabei jedoch sehr verschwenderisch mit unserem Speicherplatz umgehen (wie oft schreiben wir ´Durchgang´ ?), außerdem bereitet die immense Tipparbeit nicht allzu viel Vergnügen.

Aus diesem Grunde verzichten wir auf die Zeilen 101 bis 103, und rufen uns an dieser Stelle lieber die Befehle DATA und READ in die Erinnerung zurück.

EXKURS: READ DATA

Üblicherweise benutzen wir die INPUT - Anweisung, um Daten per Tastatur an ein laufendes Programm zu übergeben. Sie hat die Form INPUT Var, wobei Var eine beliebige Variable sein kann. Zur sequentiellen Eingabe mehrerer Daten kann eine ausreichende Anzahl Variablen, durch Kommata getrennt, angehängt werden: INPUT LÄNGE, BREITE, HÖHE kann somit drei Zahlen nacheinander zur weiteren Verarbeitung ins Programm übernehmen.
Sind die Daten bereits bei Programmstart bekannt, können sie direkt ins Programm eingebaut werden, was natürlich nur sinnvoll ist, wenn diese Daten für jeden Programmlauf stets gleich sind. Die dazu notwendigen Schlüsselworte lauten READ und DATA. INPUT wird durch READ ersetzt: READ LÄNGE, BREITE, HÖHE leistet daher die gleiche Funktion, die

Eingabedaten werden mit DATA 1,2,3 in einer beliebigen Programmzeile bereitgestellt.
Wichtig zu wissen ist, daß die Daten in der auftretenden Reihenfolge eingelesen werden, die Zahl der Daten in DATA - Zeilen muß daher gleich der Zahl der Variablen in READ - Anweisungen sein, andernfalls tritt eine Fehlermeldung auf bzw. werden nicht alle Daten berücksichtigt.
Sollen die Daten wieder mit dem ersten Element beginnend eingelesen werden, so ist der Befehl RESTORE zu verwenden.

Wir geben daher folgende Zeilen in unseren 64´er ein:

```
200 REM  -----------  RAUMBESCHREIBUNGEN
201 DATA"IM WALD."
202 DATA"IM WALD."
203 DATA"IM WALD VOR EINEM FELSHANG."
204 DATA"AUF EINER WALDLICHTUNG."
205 DATA"AUF EINER LICHTUNG AM RANDE
 EINES BERGHANGES."
206 DATA "AM EINGANGSSTOLLEN EINES ALTEN
 BERGWERKES."
```

Nun müssen die Daten an die Arbeitsvariablen überwiesen werden. Diese Zuweisung erfolgt durch READ RAUM$(I), wobei I natürlich nacheinander die Raumnummern eins bis sechs annehmen muß. Um Fehler zu vermeiden, muß die Anzahl der Räume bekannt sein; wir führen die Variable AR = Anzahl Räume ein. Zweckmäßigerweise sollte sie gleich zu Beginn des Programmtextes initialisiert werden, damit spätere Erweiterungen des Adventures leicht durchgeführt werden können.

```
110 AR=6
```

Innerhalb einer Schleife werden die Raumbeschreibungen eingelesen:

```
845 FOR RAUM=1 TO AR
850 READ RAUM$(RAUM)
870 NEXT RAUM
```

Ebenso wird unsere Richtungstabelle implementiert; um die Übersicht zu behalten, schreiben wir die Richtungswerte in die jeweiligen Datazeilen direkt hinter die Raumbeschreibungen:

```
200 REM  ----------- RAUMBESCHREIBUNGEN
201 DATA"IM WALD.",0,0,0,2,0,0
202 DATA"IM WALD.",0,0,1,3,0,0
203 DATA"IM WALD VOR EINEM FELSHANG.",0,
4,2,0,0,0
204 DATA"AUF EINER WALDLICHTUNG.",3,0,5,
0,0,0
205 DATA"AUF EINER LICHTUNG AM RANDE
 EINES BERGHANGES.",0,0,6,4,0,0
206 DATA "AM EINGANGSSTOLLEN EINES ALTEN
 BERGWERKES.",0,0,0,5,0,0
```

Die möglichen Auswege müssen daher gemeinsam mit den Beschreibungen eingelesen werden. Da auf jeden Raum sechs Richtungsdaten folgen, wird eine zweite Schleife innerhalb der ersten aufgebaut:

```
855 FOR  RICHTUNG=1 TO 6
860 READ DURCHGANG(RAUM,RICHTUNG)
865 NEXT RICHTUNG
```

Somit ist jede Reihe unseres Arrays mit einem Raum identisch; in den sechs Spalten einer jeden Reihe ist der Raum gespeichert, den der Spieler durch Eingabe der betreffenden Richtung betreten kann.

Durch diese Anordnung wird uns die im folgenden zu programmierende Fortbewegung des Spielers auf sehr einfache Art und Weise möglich sein.
Bevor dieser sich aber überhaupt irgendwohin wenden kann, muß er sich über die ihm zur Verfügung stehenden Auswege informieren können. Wir müssen also einige Programmzeilen entwickeln, die die freien Himmelrichtungen im Klartext auf dem Bildschirm ausdrucken.
Nehmen wir einmal an, unsere Hauptperson befindet sich augenblicklich in Raum 3.
Dann wird die Reihe 3 unserer Richtungstabelle angesprochen:

RAUM	*N*	*S*	*W*	*O*	*OB*	*U*
1	0	0	0	2	0	0
2	0	0	1	3	0	0
3	***0***	***4***	***2***	***0***	***0***	***0***
4	3	0	5	0	0	0
5	0	0	6	4	0	0
6	0	0	0	5	0	0

Alles was wir tun müssen, ist, die Bezeichnungen derjenigen Spalten auszugeben, die keine 0 enthalten; in unserem speziellen Beispiel die Spalten zwei und drei, Süden und Westen.

Wie wir festgestellt haben, ist die Reihe des Feldes mit dem zu untersuchenden Raum identisch; zur Bereithaltung

dieser aktuellen Raumnummer hatten wir zuvor die Variable SPIELER eingeführt. Zur Ausgabe der nicht versperrten Richtungen muß unser Programm somit die sechs Richtungen des Raumes SP testen, und die Namen aller Spalten ausdrucken, welche einen Wert ungleich Null haben:

Sechs Richtungen müssen überprüft werden:

```
1250 FOR RICHTUNG=1 TO 6
1260 IF DURCHGANG(SPI,RICHTUNG)<>0 THEN
PRINT"** AUSGANG **"
1310 NEXT RICHTUNG
```

Bei einem Probelauf wird unser Programm jetzt jeden Raum beschreiben, sowie jeden möglichen Ausgang anzeigen; leider aber noch nicht dessen Richtung. Deshalb bereiten wir ein weiteres Feld vor, ein Feld mit den Bezeichnungen der Richtungen (RICHTUNG$):

```
1020 DATA NORDEN, SUEDEN, WESTEN, OSTEN,
 OBEN, UNTEN
1030 FOR RICHTUNG=1 TO 6
1040 READ RICHTUNG$(RICHTUNG)
1050 NEXT RICHTUNG
```

In den einzelnen Elementen sind die Bezeichnungen der sechs Richtungen gespeichert, und wir können uns das Feld RICHTUNG$ als eine Art Schablone vorstellen, die während des Programmlaufs auf die gerade aktuelle Reihe unserer Richtungstabelle gelegt wird:

```
1260 IF DURCHGANG(SPIELER,I)<>0 TH
EN PRINT RICHTUNG$(RICHTUNG)
```

Zur Kontrolle starten wir unser Programm und geben einige Raumnummern ein - ein Blick auf unsere Karte wird uns davon überzeugen, daß wir bisher alles richtig gemacht haben.

Nach diesen Vorbereitungen werden wir nun die gezielte Bewegung in unserem Adventure möglich machen.
Jeder, der bereits einmal ein Abenteuerspiel geladen hatte, wird wissen, daß die Eingabe einer Himmelsrichtung zum Zwecke der Fortbewegung die wohl häufigste Anweisung an die Spielfigur ist. Daher sollten wir als Programmierer dem Spieler soweit entgegenkommen, daß wir die Eingabe des Anfangsbuchstabens als ausreichend betrachten, womit wir dem Abenteurer das Ausschreiben der Himmelrichtungen, und somit mehrere hundert Tastendrücke, ersparen.
Tatsächlich lassen sich zahlreiche Abenteuerspiele finden, die für die Fortbewegung keine von der Befehlsdecodierung und -ausführung unabhängige Routine aufweisen, und explizit ein GEH WESTLICH erfordern.
Wir wollen jedoch Richtungsanweisungen bevorzugt behandeln, was auch einer schnelleren Reaktion des Programmes auf die Eingabe hin zugute kommt.

Ersetzen wir zunächst die Zeile 1390, um adventuregemäße Eingaben zu ermöglichen:

```
1390 INPUT "WAS SOLL ICH TUN";EINGABE$
```

Bevor unser Programm nun irgendwelche Manipulationen des Spieles durchführt, sollte es zunächst die Länge der Spielereingabe überprüfen. Ist die Eingabe länger als zwei Buchstaben (oben = OB), wird es sich um irgendeine Handlung handeln, andernfalls muß die Bewegungsroutine durchlaufen werden. Diese testet zunächst, ob der Weg in der gewünschten Richtung überhaupt frei ist, wenn ja, wird in der Richtungstabelle unter der entsprechenden Richtung (Spalte) des Raumes, in welchem der Spieler sich aufhält (Reihe), der neue Raum abgelesen und der entsprechenden

Variablen (Spieler) zugewiesen. Abschließend wird der Spieler über die Durchführung der Aktion informiert:

```
1080 PRINT
1400 IF LEN(EI$)>2 THEN 1500
1410 IFEI$="N"ANDDURCHGANG(SPI,1)<>OTHEN
SPI=DURCHGANG(SPI,1):PRINT"O.K.":GOTO108
O
1420 IFEI$="S"ANDDURCHGANG(SPI,2)<>OTHEN
SPI=DURCHGANG(SPI,2):PRINT"O.K.":GOTO108
O
1430 IFEI$="W"ANDDURCHGANG(SPI,3)<>OTHEN
SPI=DURCHGANG(SPI,3):PRINT"O.K.":GOTO108
O
1440 IFEI$="O"ANDDURCHGANG(SPI,4)<>OTHEN
SPI=DURCHGANG(SPI,4):PRINT"O.K.":GOTO108
O
1450 IFEI$="OB"ANDDURCH(SPI,5)<>OTHENSPI
=DURCHGANG(SPI,5):PRINT"O.K.":GOTO108
O
1460 IFEI$="U"ANDDURCHGANG(SPI,6)<>OTHEN
SPI=DURCHGANG(SPI,6):PRINT"O.K.":GOTO108
O
1470 PRINT"DAHIN FUEHRT KEIN WEG !":GOTO
 1080
1499 REM SPAETER EINGABEANALY.
```

Mit Beginn eines jeden Spieles müssen wir dem Programm natürlich den Startraum mitteilen:

```
150 SPIELER=1
```

Ein kurzer Spaziergang durch unsere Adventurewelt wird uns nun schnell davon überzeugen, daß wir als nächstes unbedingt die Gestaltung des Monitorbildes in Angriff nehmen müssen, oder unseren Augen wird sich nach einigen Spielzügen ein wenig informatives Durcheinander bieten.

FORMATIERUNG DER AUSGABE

Um eine saubere Bildschirmgestaltung zu ermöglichen, muß mit Spielstart der Bildschirm natürlich gelöscht werden:

```
1070 PRINT CHR$(147)
1080 PRINT
```

Diese Leerzeile ist auf einem leeren Schirm selbstverständlich sinnlos. Erforderlich wird sie immer nach Ausführung eines Befehls, denn unsere Eingabe, wie auch eventuell erfolgte Mitteilungen, müssen um eine Zeile nach oben rutschen. Dieses Scrollen erzielen wir durch ein PRINT in die letzte Bildschirmzeile und ist als Abschluß eines jeden Spielzuges vonnöten.

Für unsere Eingaben benötigen wir die unterste Zeile, uns stellt sich nun die Frage, wie wir nach Ausgabe der Informationen an den Spieler im oberen Drittel des Schirmes dorthin gelangen.

Von seinen Fähigkeiten her bietet uns der Commodore 64 zwei Möglichkeiten. Da wären beispielsweise die Cursor - Steuerbefehle. Wir könnten den Cursor um eine entsprechende Anzahl von Zeilen nach unten bewegen, doch leider ist diese Differenz von der Menge der ausgegebenen Informationen, sprich Anzahl von Gegenständen, abhängig, und dieser Wert steht während des Spieles nicht so ohne weiteres zur Verfügung.

Was wir brauchen, ist der ´PRINT AT Position´ Befehl, ein Befehl, welcher außer den zu druckenden Daten auch deren genaue Ausgabeposition auf dem Bildschirm berücksichtigt. Scheinbar läßt uns unser 64´er hier im Stich, tatsächlich bietet er uns die gleiche Möglichkeit wie der IBM-PC oder andere 16-bit Rechner. Zwar finden wir ihn nicht im Commodore Basic, aber er existiert dennoch, der sogenannte LOCATE- Befehl. Dieser Befehl positioniert den Cursor vor einem Schreibvorgang an der gewünschten Stelle, was durch

Angabe der betreffenden Zeile und der Schreibstelle innerhalb dieser Zeile geschieht.

Unser Basicinterpreter muß während einer Bildschirmausgabe natürlich genau wissen, wo der Cursor sich befindet, und hat zu diesem Zweck zwei Speicherstellen, die für die Aufnahme dieser Daten reserviert sind. Diese Eigenschaft werden wir uns zunutze machen, indem wir eben dafür sorgen, daß der Interpreter unter diesen Adressen, es handelt sich dabei um die Speicherstelle 214 für die Zeile und um Adresse 211 für die Spalte, die uns gerade passenden Daten findet. Anschließsend lassen wir ein Unterprogramm unseres Interpreters den Cursor an der gewünschten Stelle positionieren.

```
1390 POKE 214,24:POKE 211,0:SYS 58732:IN
PUT"Was soll ich tun";EINGABE$
```

Unsere Eingabe erfolgt somit in Zeile 24 und das abschließende RETURN wird den gesamten Bildschirminhalt um eine Zeile nach oben schieben, weshalb für die Mitteilung der Reaktionen an den Spieler keine weiteren Maßnahmen getroffen werden müssen. Mit Beginn des neuen Zuges macht Zeile 1080 die unterste Reihe für den nächsten Eingabezyklus frei.

Leider werden momentan alle Ausgaben in der untersten Bildschirmzeile gemacht, die Raumbeschreibungen müssen jedoch in der obersten Zeile beginnen:

```
1130 POKE 214,0 : POKE 211,0 : SYS 58732
```

Den Abenteurer über die freien Ausgänge zu informieren, wird für uns allerdings nicht ganz so einfach sein, denn unsere bisherige Verfahrensweise läßt es nicht zu, mehr als drei Wege einwandfrei auszugeben.
So sollte kein Wort über das Ende der Zeile hinausgehen, ebenso sollten die verschiedenen Himmelsrichtungen durch

ein Komma getrennt und der Satz mit einem Punkt abgeschlossen werden.

Ändern wir zunächst Zeile 1260,

```
1260 IF DURCHGANG(SPI,RICHTUNG)=0 THEN G
OTO 1310
```

und gehen wir für unsere weiteren Überlegungen von folgendem Beispiel aus:

ICH KANN NACH NORDEN, SUEDEN, OSTEN,
OBEN, UNTEN.

Der erste Ausgang macht uns keine Schwierigkeiten, er kann ohne alle Vorarbeit ausgedruckt werden. Wir müssen diese erste Ausgabe nur von allen weiteren unterscheiden, was auch ohne weiteres möglich ist. So wird die erste Richtungsangabe immer nach ´Ich kann nach ´ an der vierzehnten Schreibposition erscheinen, deshalb:

```
1270 IF POS(0)=14 THEN PRINT RICHTUNG$(R
ICHTUNG);:GOTO 1310
```

Für alle folgenden Richtungsangaben müssen wir zunächst jedoch einen Test durchführen. Es ist wichtig zu wissen, ob wir auch nicht über das Ende der Zeile hinaus schreiben. Sollte dies nicht der Fall sein, drucken wir zunächst ein Komma und dann die nächste Himmelsrichtung:

```
1280 IFPOS(0)+LEN(RI$(RICHTUNG))<37 THEN
PRINT", ";RI$(RICHTUNG);:GOTO 1310
```

Sollte die Gesamtausgabelänge größer als 37 werden, erfolgt die Ausgabe eines Kommas sowie die Ansteuerung der nächsten Zeile mittels eines PRINT- Befehles:

```
1290 IF POS(0)+LEN(RI$(RICHTUNG))>=37THE
N PRINT",":PRINT RI$(RICHTUNG);:GOTO1310
```

Damit wäre unsere Beispielzeile bis auf ´UNTEN´ gedruckt. Diese Richtungsangabe liegt in einem Zeilenbereich, der durch unsere Bedingungen noch nicht abgefragt wird. Typisch für ihn ist, daß er eine Schreibposition größer zwei und kleiner sechzehn haben muß:

```
1300 IF POS(0)<16 AND POS(0)>2 THEN PRIN
T", ";RICHTUNG$(RICHTUNG);:GOTO 1310
```

Mit diesen vier Zeilen haben wir alle auftretenden Möglichkeiten abgedeckt. Der abschließende Punkt ist auf jeden Fall nur ein einziges Mal erforderlich und wird deshalb nach Verlassen der Schleife gesetzt:

```
1310 NEXT RICHTUNG
1320 PRINT"."
```

Vielleicht bedarf es noch einer Erklärung, warum wir die Zeilenlänge nur auf kleiner 37 testen, wo eine Bildschirmzeile doch von 0 bis 39 geht. Nun, die Antwort finden Sie sowohl in Zeile 1320 als auch im Komma in 1280.

GOLD, SILBER UND ANDERE NÜTZLICHE DINGE

Eine Wanderung durch unsere Adventurewelt verschafft dem Abenteurer bislang einen recht trostlosen und uninteressanten Eindruck, stehen ihm doch keinerlei Handlungsgegenstände zur Verfügung. Ebenso werden reine Ortsbeschreibungen beim besten Willen kein Spiel ermöglichen. Nehmen wir also wieder einmal unsere Karte zur Hand und bereichern wir unsere Welt um die gewünschten Objekte, wobei wir uns in der Regel auf für den Fortlauf des Spieles wichtige Gegenstände beschränken werden. Denn für reine Verschönerungen wäre der spätere Aufwand zu groß, schließlich müssen wir gewährleisten, daß der Spieler alles das, was er findet, auch in die Hand nehmen und untersuchen kann. Dennoch kommen wir nicht umhin, in jedem Raum mindestens ein Objekt zu plazieren, denn daß der Abenteurer so ohne weiteres gar nichts sieht, ist doch sehr zweifelhaft. Falls die Phantasie nicht ausreicht, ist das betreffende Objekt eben *nichts besonderes*, eine Formulierung die wir auch deshalb so häufig in Adventurespielen finden, weil sie gerne zur Verminderung des Arbeitsaufwandes gewählt wird, denn in einem normalem Wald sind Bäume, Sträucher, Gras, Insekten usw. eben wirklich nichts besonderes.
Oftmals kann jedoch auch auf unnötige Dinge nicht verzichtet werden, besonders dann nicht, wenn sie zum Milieu gehören und das Adventure einen realistischen Eindruck hinterlassen soll.

So werden auch wir in unserem Wald Bäume aufstellen müssen, obwohl die Handlung sie nicht erfordert. In Raum zwei, der bereits nahe dem Gebirge liegt, vertiefen einige Felsbrocken den Eindruck der Wirklichkeit. Weiterhin gehört zu einem Bergwerk eine Baracke, egal ob es nun die Wohnstätte des Minenbesitzers oder ein Geräteschuppen ist. Irgendwo müssen wir schließlich auch die zu suchenden Schätze verstecken, zusätzlich sind einige Gegenstände, die dem Spieler eher schaden als nützen, angebracht.

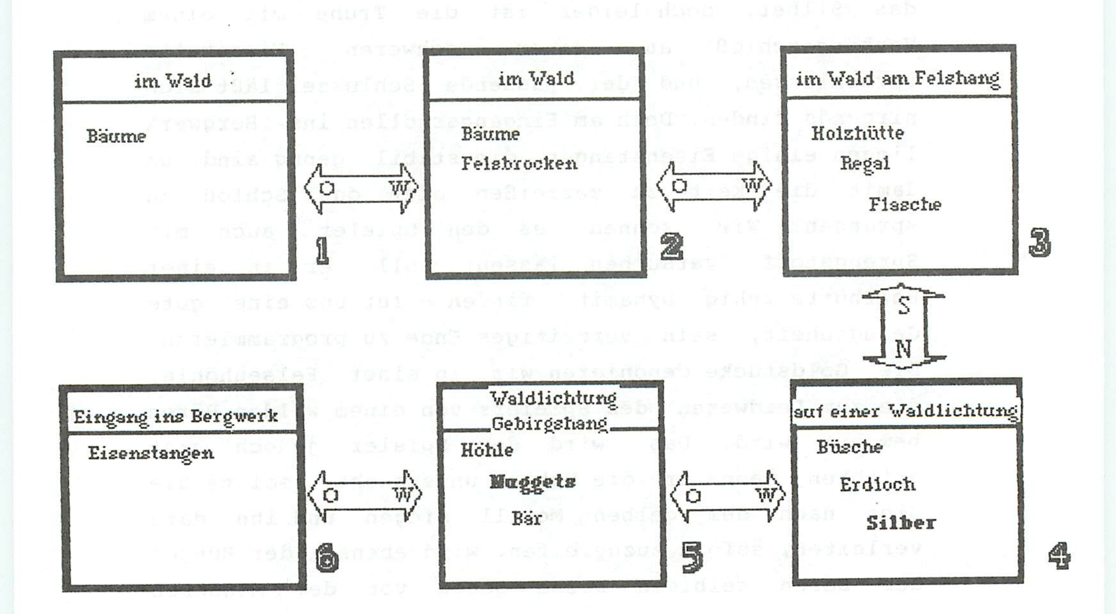
Lassen Sie mich daher bitte im folgenden den mir vorschwebenden Handlungsablauf näher konkretisieren und die benötigten Gegenstände auswählen:

HANDLUNGSABLAUF GOLDRAUSCH

Version 1 soll dem Spieler die Aufgabe stellen, zwei Schätze, Goldstücke und Silbermünzen, im Umfeld der Mine zu finden. Bei Spielbeginn soll sich der Spieler im Wald befinden und sich zunächst an das Bergwerk herantasten müssen. Auf dem Weg zur Mine entdeckt er im Wald ein Erdloch, auf dessen Grund sich eine schwere Eisentruhe befindet. In dieser befindet sich das Silber, doch leider ist die Truhe mit einem Vorhängeschloß an einer schweren Eisenkette verschlossen, und der passende Schlüssel läßt sich nirgends finden. Doch am Eingangsstollen ins Bergwerk liegen einige Eisenstangen, die stabil genug sind, um damit die Kette zu zerreißen oder das Schloß zu sprengen. Wir können es den Spieler auch mit Sprengstoff versuchen lassen, soll er in einer Holzhütte ruhig Dynamit finden - für uns eine gute Gelegenheit, sein vorzeitiges Ende zu programmieren. Die Goldstücke deponieren wir in einer Felsenhöhle, die zum Leidwesen des Spielers von einem wilden Bären bewohnt wird. Das wird der Spieler jedoch erst erfahren, wenn er die Höhle untersucht, sollte die Gier nach dem gelben Metall siegen und ihn dazu verleiten, sofort zuzugreifen, wird ebenso der Hunger des Bären selbigen seine Scheu vor dem Menschen verlieren lassen - nach dem Dynamit die zweite Falle für den Abenteurer.

Andererseits wissen wir alle dank Wilhelm Busch, daß man Bären mit Honig eine große Freude machen kann, stellen wir auf einem Regal in der Hütte daher eine Flasche bereit, die den begehrten Stoff enthält. Betritt der Spieler damit gewappnet die Höhle, wird der Bär den Honig wittern und mit der Flasche in den Tiefen der Höhle verschwinden, womit einem erfolgreichem Ende dieses Miniadventures nichts mehr im Wege steht.

Die Karte zu unserem Adventure *Goldrausch* dürfte nun so oder ähnlich aussehen:



Bevor wir mit der Programmierung der Objekte unser Adventure vervollständigen, sind einige weitere Vorüberlegungen notwendig.

Um dem Spieler das Hineinleben in unsere Computerwelt zu ermöglichen, werden die Beschreibungen dieser Welt nicht aus trockenen Substantiven bestehen, sondern wir werden das Interesse des Spielers durch plastische Schilderungen wachhalten. Die lapidare Mitteilung ´Ich sehe einen Bären.´ läßt den Spieler unser Adventure zwar spielen, die Nachricht ´Ich sehe einen äußerst grimmig dreinblickenden Bären´, wird ihn unser Adventure dagegen erleben lassen.

Hat der Spieler nun das Bestreben, einen neuen Freund zu finden, wäre eine Eingabe wie ´Streichel äußerst grimmig dreinblickenden Bären´ für uns als Programmierer zwar einfach zu realisieren, für den Spieler aber unzumutbar.
Daher werden wir jedem Objekt neben der eigentlichen Objektbeschreibung einen Rufnamen zuordnen. So erhalten wir auch die Möglichkeit, die Wortlänge unseres Adventures nach Bedarf frei festzulegen. Vermutlich haben Sie bemerkt, daß professionelle Adventures aus dem englischsprachigen Raum meist nur die ersten drei oder vier Buchstaben zur Identifikation einer Handlung erfordern. Drei Buchstaben sind auch für deutsche Adventures in der Regel ausreichend, nur bei der Realisation eines größeren Projektes sollten Sie zuvor Ihre Wortliste aufmerksam durchsehen, sonst werden Sie bei der Programmierung der Bedingungen und Aktionen überrascht feststellen, wie viele für ein Adventure wichtige Substantive mit Sch beginnen. Wird Ihr Spiel weiterhin mit Groß- und Kleinschrift ausgestaltet, achten Sie bitte darauf, bei den Rufnamen nur Großbuchstaben zu verwenden, denn diese Kürzel werden später mit den Eingaben des Spielers auf Gleichheit überprüft.

Diese zwei Felder, OB$ für die Objektbeschreibungen und RN$ für die Rufnamen, werden durch ein drittes Feld OB ergänzt, in dem die Nummer des Raumes gespeichert wird, in welchem das

betreffende Objekt sich gerade befindet. Dadurch haben wir ein einfaches Mittel zur Hand, die Position eines jeden Gegenstandes zu kontrollieren und zu manipulieren. Solange ein Gegenstand nicht aktiv am Spiel beteiligt ist, ist OB = 0, andernfalls entspricht der Inhalt von OB dem Wert der Variablen Spieler, oder ist gleich -1 in dem Fall, daß der Abenteurer ihn mit sich führt.

Die folgenden DATA - Zeilen beinhalten alle Gegenstände, die für eine Realisation des bisherigen Spielplanes (Goldrausch, Version 1) erforderlich sind. Die Zahl am Ende der jeweiligen Zeile gibt an, in welchem Raum sich das Objekt bei Programmstart befindet. Die Anführungsstriche sind übrigens nur dann unbedingt erforderlich, wenn zur Objektbeschreibung selbst ein Komma gehört.

```
300 REM -------------------- GEGENSTAENDE
301 DATA"VIELE GROSSE BAEUME","BAE",1
302 DATA"VIELE GROSSE BAEUME","BAE",2
303 DATA"EINIGE FELSBROCKEN","FEL",2
304 DATA"EINE VERFALLENE HOLZHUETTE","HU
E", 3
305 DATA"EINE VERSCHMUTZTE KORBFLASCHE",
"FLA",0
306 DATA"HONIG","HON",0
307 DATA"EINE HOLZKISTE", "KIS",3
308 DATA"EIN KLAPPRIGES REGAL","REG",0
309 DATA"ETWAS SPRENGSTOFF", "SPR",0
310 DATA"EIN DUESTERES ERDLOCH","ERD",0
311 DATA"EINE ROSTIGE EISENTRUHE","TRU",
0
312 DATA"*SILBERMUENZEN*", "SIL", 0
313 DATA"EINE DUESTERE FELSENHOEHLE","HO
E",5
314 DATA"EINEN GRIMMIG DREINBLICKENDEN B
AEREN", "BAE", 0
```

```
315 DATA"ZAHLLOSE NIEDRIGE BUESCHE","BUE
", 4
316 DATA"MEHRERE EISENSTANGEN","EIS",6
317 DATA"*NUGGETS*", "NUG", 5
```

Genau wie bei den Raumbeschreibungen müssen wir vor Initialisierung der Variablen für Objektbeschreibungen (OB$), Rufnamen (RN$) und Positionen (OB) auch hier wieder die Anzahl der Objekte festlegen:

```
120 AO=17
830 FOR OBJEKT=1 TO AO
835 READ OB$(OBJEKT), RN$(OBJEKT), OB(OB
JEKT)
840 NEXT OBJEKT

190 DIM RAUM$(AR), DURCHGANG(AR,6), OB$(
AO), RN$(AO), OB(AO)
```

Zur Speicherung aller Objekte sind 17 indizierte Variablen notwendig, für die der benötigte Speicherplatz durch die DIMensionierungsanweisung in Zeile 190 bereitgestellt wird. Wie die meisten Computer, so reserviert auch unser Commodore mit dem Start von Basic elf Elemente pro Liste, weshalb wir trotz des Fehlens dieser Zeile bei der Initialisierung der Orte keine Fehlermeldung erhalten hatten.

Wie machen wir die Einrichtung eines Raumes dem Spieler sichtbar ? - Nun, ganz einfach: die Variable SPIELER enthält die Nummer des betretenen Raumes, OB() enthält die Raumnummern der jeweiligen Objekte. Wir testen innerhalb einer Schleife für alle Gegenstände, ob diese Positionsnummer gleich der aktuellen Raumnummer ist:

```
1160 PRINT"ICH SEHE ";
```

```
1170 FOR I=1 TO AO
1180 IF OB(I)<>SPIELER THEN 1210
1190 IF POS(0)+LEN(OB$(I))+2<=39 THEN PR
INT OB$(I);", ";:GOTO 1210
1210 NEXT I
```

Programmzeile 1180 testet für jeden Gegenstand, ob dieser sich in einem anderen Raum als der Spieler befindet, fällt dieser Test positiv aus, wird sofort das nächste Objekt überprüft. Ansonsten wird die betreffende Objektbeschreibung in Zeile 1190 ausgegeben. Das Semikolon sorgt dafür, daß nicht für jedes weitere Objekt eine neue Ausgabezeile benutzt wird. Zur Vermeidung einer Ausgabe über die Zeile hinaus ergänzen wir:

```
1200 IF POS(0)+LEN(OB$(I))+2>39 THEN PRI
NT : GOTO 1190
```

Vor Ausgabe des nächsten Gegenstandes wird die Gesamtlänge des Ausdrucks errechnet. Wenn der Cursor durch Drucken der Gegenstandsbeschreibung, des Kommas und der Leerstelle zwischen zwei Objekten (daher +2) eine Position größer Spalte 39 erreichen würde, wird ein Zeilenvorschub ausgelöst. Danach verzweigt das Programm wieder zur Zeile 1190, die Bedingung ist nun erfüllt und die Beschreibung wird ausgedruckt.
Hinter dem letzten Objekt wirkt das Komma jedoch unschön, weshalb wir den Cursor um zwei Positionen nach links bewegen, und einen Punkt über das Komma schreiben:

```
1220 PRINT CHR$(157);CHR$(157);"."
```

Um den Bildschirmaufbau der amerikanischen Originaladventures zu erreichen, fehlen uns noch eine Trennungslinie sowie eine weitere Leerzeile:

```
1010 LEERZEILE$="
                  "
1230 PRINT LEERZEILE$
```

```
1330 PRINT"------------------------------
----------"
```

Ein letztes Problem werden wir wiederum nach mehreren Eingaben feststellen. Die Mitteilungen innerhalb der unteren Schirmhälfte rücken immer weiter nach oben und vermischen sich nach einigen Zügen mit den Beschreibungen der Umgebung. Es ist Aufgabe des Programmes, vor deren Ausgabe die obersten Zeilen zu löschen, was durch Drucken einer ausreichenden Menge von Leerzeilen geschehen kann:

```
1090 POKE 211,0: POKE 214,0: SYS 58732
1100 FOR ZEILE=1 TO 10
1110 PRINT LEERZEILE$
1120 NEXT ZEILE
```

Bevor wir nun das eigentliche Spiel programmieren und uns mit den für einen Spieler nicht sichtbaren Aktionen des Programmes beschäftigen, müssen wir schnell noch die dazu notwendigen Verben eingeben. Neben einer Reihe von immer erforderlichen Verben wie *Untersuche*, *Nimm*, *Leg* werden auch sie vom Inhalt des jeweiligen Adventures abhängen. Die Implementation erfolgt entsprechend den Räumen und Objekten, wobei eine der Wortlänge des Adventures entsprechende Abkürzung ausreichend ist:

```
130 AV=7
500 REM ------------------------ VERBEN
501 DATA UNT, NIM, LEG, OEF, BEN, ERS, V
ER
860 FOR I=1 TO AV
865 READ VERB$(I)
870 NEXT I
```

Weiterhin können wir mit einigen wenigen Ergänzungen das Bild attraktiver gestalten. Unterschiedliche Farben werden uns helfen, die während des Spieles vielfältigen Mitteilungen zu differenzieren. Ich habe mich für einen schwarzen Hintergrund, auf dem die Mitteilungen der Hauptperson an uns in hellem Blau ausgegeben werden, entschieden. Die freien Wege werden in orange ausgedruckt und unsere Eingaben in weiß gemacht. Natürlich steht es Ihnen frei, andere Steuercodes, entsprechend Ihrem Geschmack, zu verwenden.

```
10 PRINT CHR$(147)
11 POKE 53280,12:POKE 53281,11
12 PRINT CHR$(154)
1330 PRINT CHR$(5);"------------------
---------------------"
1390 POKE 211,0:POKE 214,24:SYS 58732:PR
INTCHR$(5);:INPUT"WAS SOLL ICH TUN";EI:P
RINTCHR$(155);
```

Damit haben wir die im ersten Kapitel dieses Buches beschriebenen äußere Gestaltung erreicht und werden nun das Spielverhalten unseres Programmes festlegen.

ANALYSE DER EINGABEN

Vom Spieler gewünschte Richtungsänderungen, eine Befehlsgruppe mit Wortlängen von ein bis zwei Buchstaben, werden bereits erkannt und korrekt ausgeführt. Die Standardeingabe besteht jedoch immer aus Verb und Objekt, wobei die Wortlängen, abgesehen von der zur Erkennung der Begriffe notwendigen Mindestwortlänge, keinen Restriktionen unterworfen werden. Um wie vom Spieler gewünscht reagieren zu können, muß unser Programm dessen Eingabe überprüfen:

- ist das benutzte Verb programmiert ?
- ist das Objekt vorgesehen ?

Können beide Fragen mit ja beantwortet werden, wird die Eingabe in Verb und Objekt zerlegt, Verb- und Objektnummer werden ermittelt; sind die Bestandteile der Eingabe nicht definiert, wird eine entsprechende Mitteilung an den Abenteurer ausgegeben.

EXKURS: STRINGBEHANDLUNG

Der wesentliche Unterschied zwischen einem Computer und einem programmierbaren Taschenrechner ist die Fähigkeit des Computers, mit Texten umzugehen. Natürlich hat ein Rechner bislang kein Verständnis für diese Texte, sondern er faßt sie als eine Ansammlung bestimmter Zeichen, als sogenannte Zeichenkette oder String auf. Bei diesen Strings handelt es sich im allgemeinen um Kombinationen aller per Tastatur eingebbaren Zeichen. Das Betriebssystem des Computers stellt nun eine Reihe von Funktionen zur Behandlung dieser Zeichenketten zur Verfügung.
Bei unserem Commodore 64 sind dies neben einer Reihe von Konvertierungsbefehlen wie VAL() und STR$(), die eine Zahl

in einen String, bzw. einen String in eine Zahl verwandeln, die Befehle LEFT$, RIGHT$ und MID$.
Diese drei Funktionen stellen zur weiteren Bearbeitung durch das Programm einen genau definierten Teilstring aus der gesamten Zeichenkette zur Verfügung. So ergibt die Funktion LEFT$("ABCDEF",3) eine Kette der linken drei Zeichen, also den Substring ABC. Als notwendige Parameter müssen mit dem Aufruf der Funktion also zunächst die zu bearbeitende Zeichenkette und die Anzahl der gewünschten Buchstaben übermittelt werden. Entsprechendes gilt für RIGHT$. MID$(X$,S,X) stellt uns beginnend ab Position S einen String X Zeichen lang von X$ zur Verfügung. So wird unser Adventureprogramm beispielsweise die Position des Leerzeichens zwischen Verb und Objekt ermitteln, und diese dann mit LEFT$ und RIGHT$ aus EINGABE$ des Spielers ermitteln. Um die Anzahl erforderlicher Buchstaben jedoch berechnen zu können, muß zuvor mit LEN(EINGABE$) die Gesamtlänge der Eingabe ermittelt werden.

Wird die Ausführung einer Eingabe nicht in den Zeilen 1290 bis 1340 durchgeführt, so wird, falls die Länge der Eingabe weniger als drei Buchstaben beträgt, angenommen, daß es sich um einen Eingabefehler handelt. Nachdem *Dahin führt kein Weg!* ausgegeben worden ist, beginnt ein neuer Spielerzug.
Handelt es sich um eine längere Eingabe, wird der Programmlauf mit Zeile 2000 fortgesetzt:

```
1470 IF LEN(EINGABE$)<3 THEN PRINT
"DAHIN FUEHRT KEIN WEG !": GOTO 1080

2000 LN=LEN(EINGABE$)
2010 FOR EL=1 TO LN
2020 TEST$=MID$(EINGABE$,EL,1)
2030 IF TEST$<>" "THEN NEXT EL
```

Nach Ermittlung der Länge der gesamten Eingabe wird innerhalb einer Schleife jeder Buchstabe, beginnend von links, darauf überprüft, ob er mit einem Leerzeichen identisch ist. Ist das Leerzeichen gefunden, steht die Länge des Verbes fest (bisher geprüfte Zeichen minus 1) und das Verb kann der Variablen EV$ zugewiesen werden. Nach Ermittlung der Restlänge kann das Objekt an die Variable EO$ übergeben werden.
Wurde ein ´Ein Wort Befehl´ verwendet (Hilfe), so wird die Restlänge kleiner Null sein. Da kein Objekt zur weiteren Bearbeitung bereitgestellt werden muß, wird direkt (Zeile 2060) zur Verbanalyse (Zeile 2090) verzweigt:

```
2040 EV$=LEFT$(EINGABE$,EL-1)
2050 RL= LN-EL
2060 IF RL<O THEN 2090
2070 EO$=RIGHT$(EINGABE$,RL)
2090 FOR VN=1 TO AV
2100 IF EV$=VERB$(VN) THEN 2130
2110 NEXT VN
```

Innerhalb einer Schleife wird der Text des ersten eingegeben Wortes mit den im Spiel möglichen Verben verglichen. Wird ein identischer String gefunden, entspricht der Schleifenindex der Verbnummer, und die Schleife wird verlassen. Ist die Schleife vollständig abgearbeitet und keine Identität festgestellt worden, so hatte der Programmierer das betreffende Verb nicht vorgesehen, worüber dem Spieler Mitteilung gemacht wird:

```
2120 PRINT "DAS VERB VERSTEHE ICH NICHT"
!" : GOTO 1080
```

Anschließend wird nach gleichem Prinzip die Objektnummer ermittelt:

```
2130 FOR N=1 TO AO
2140 IF EO$=RN$(N) THEN 5000
```

```
2150 NEXT N
2160 PRINT "ICH VERSTEHE DAS OBJEKT NICH
T !": GOTO 1080
```

Mit Eingabe dieser Zeilen haben wir einen wesentlichen Teil unserer Entwicklungsarbeit hinter uns gebracht. Obige Routinen sorgen für einen ansprechenden Aufbau des Bildschirmes, wie sie auch das ´Verständnis´ für die Eingabe des Spielers erzeugen.
Sie ermitteln, welches der möglichen Verben der Abenteurer benutzt hat und mit welchem Gegenstand er etwas tun möchte, womit eine Verzweigung des Programmablaufes an eine zur Ausführung der betreffenden Aktion vorgesehene Programmzeile möglich wird. Wegen dieser zentralen Steuerungsaufgaben wird dieser Programmteil auch als TREIBER bezeichnet.

ÜBERBLICK: AUFBAU DER PROGRAMME

Somit wird deutlich, daß unsere Abenteuerprogramme im wesentlichen aus drei Teilen bestehen:

1. *Daten des Adventures*
2. *Adventuretreiber*
3. *Ausführung der Spielzüge*

1. Die Daten als Grundlage des Spieles werden nach Programmstart an die Arbeitsvariablen übergeben.

2. Dem Treiber fällt die Steuerung des gesamten Programmablaufes zu. Er gestaltet die Bildschirmausgabe, übernimmt die Eingaben des Spielers, wertet sie aus und leitet die Befehlsausführung ein.
Der Treiber ist vom jeweiligen Adventure unabhängig und wird unverändert für alle Adventures übernommen.

3. Der dritte Programmteil stellt das eigentliche Adventure dar. Treffen alle für eine Handlung notwendigen Bedingungen zu, wird diese Aktion ausgeführt.

Um die Übersicht zu behalten, wird der Aufbau unserer Textadventures schematisch dargestellt, wobei die zur Verfügung stehenden Zeilennummern mit angegeben werden.

SCHEMATISCHE DARSTELLUNG: TEXTADVENTURE

INITIALISIERUNG

100	*200*	*KONTROLLVARIABLEN INITIALISIEREN*
200	*300*	*VERBEN*
300	*500*	*OBJEKTBESCHREIBUNGEN*
		LAGE DER OBJEKTE
500	*600*	*RAUMBESCHREIBUNGEN*
		VERBINDUNGEN DER RÄUME
600	*700*	*MITTEILUNGEN AN DEN SPIELER*
900	*1000*	*SPIELVARIABLEN INITIALISIEREN*

ADVENTURETREIBER

1000	*1070*	*INITIALISIERUNG RICHTUNGEN*
1080	*1330*	*BILDSCHIRMVERWALTUNG*
1390		*EINGABE DES SPIELZUGES*
1400	*1500*	*HAUPTPERSON BEWEGEN*
2000	*3000*	*ANALYSE DER EINGABE*

AUSFÜHRUNG

5000 30000 SPIELZÜGE

Für die später zu erstellenden Erweiterungen stehen in den einzelnen Blöcken genügend nicht benutzte Zeilennummern zur Verfügung.

Wir werden sehen, wie dieses Konzept leicht zu einem Adventureinterpreter ausgebaut oder auch zur Realisierung von Grafikadventures eingesetzt werden kann.

So werden wir den Treiber fast unverändert übernehmen können, und müssen nur die übrigen Programmteile modifizieren.

WANN GEHT WAS ?

Stellen wir uns nun den Spieler vor, der unsere zwei Schätze finden will. Mit einem Blick stellt er fest, daß er sich mitten in einem Wald befindet, umgeben von Bäumen und Felsbrocken. Als geübter Adventurespieler oder Leser der vorangegangenen Kapitel, weiß er, daß des Pudels Kern in den unscheinbarsten, alltäglichsten Dingen liegen kann.

Wie wird er reagieren ?
Welche Eingaben müssen wir erwarten ?

Mit an Sicherheit grenzender Wahrscheinlichkeit probiert er mit Anweisungen wie *UNTERSUCHE WALD, UNTERSUCHE BAUM, oder UNTERSUCHE FELSBROCKEN* weiterzukommen.
NIMM BAUM ist weniger wahrscheinlich, dennoch müssen wir, um allen Kritikern unserer Programme den Wind aus den Segeln zu nehmen, eine Mitteilung der Art *So stark bin ich nicht*. programmieren, außerdem animieren vielfältige Mitteilungen, die zeigen, daß eine Eingabe auch wirklich verstanden wurde, den Abenteurer zum Weiterspielen.
Verständlich - wäre das Ergebnis jeder zweiten oder dritten Eingabe ein *Ich verstehe nicht was Du meinst !* dann wird der Reiz zum Abschalten des Computers doch sehr groß.

In der Praxis werden wir für die Räume 1 und 2 daher, weil wir hier noch keine Handlung vorgesehen haben, sondern nur unsere Welt größer wirken lassen wollen, nur die Ausgabe diverser Meldungen an den Spieler vorsehen, eine Aufgabe, die ein Print - Befehl schnell erledigt. Doch nachdem der Spieler Raum 3 betreten und die Holzhütte entdeckt hat, wird er diese ebenfalls in Augenschein nehmen und dabei das Regal mit der Korbflasche entdecken. Diese muß natürlich ab diesem Zeitpunkt, zusätzlich zum Regal und zur Hütte, in der Beschreibung der Szene erscheinen ('Ich sehe ...'), das heißt, wir müssen ihren Aufenthaltsort verändern. Weiteres Kopfzerbrechen könnte uns die Eingabe *NIMM FLASCHE*

bereiten: nun muß die soeben erschienene Korbflasche wieder verschwinden. Zurück an ihren alten Platz kann sie nicht, denn wenn unser Spieler *INVENTUR* macht, muß sie selbstverständlich aufgelistet werden.

Ebenso selbstverständlich kann der Spieler die Flasche nicht schon in Raum 1 untersuchen, genau so wenig in Raum 2, überhaupt sollten alle Eingaben betreffend der Flasche abschlägig beantwortet werden, solange sie sich nicht in unmittelbarer Nähe des Spielers befindet.

Dies ist jedoch eine Geschmacksache, die im Ermessen des Programmierers liegt. Denn dieser kann die Meinung vertreten, daß die Flasche in Wirklichkeit bereits seit undenklichen Zeiten unberührt in dem Regal steht, und daß der Spieler sie nur noch nicht gesehen hat.
Wird dieser Spieler durch die zahlreichen Überraschungen eines Abenteuers jedoch gezwungen, noch einmal von vorne zu beginnen, kennt er die Startpositionen diverser Gegenstände, und man kann es ihm ohne weiteres ermöglichen, diese zu nehmen, auch ohne daß sie in der Zeile ´Ich sehe ...´ gelistet worden waren.

In der Tat wird der Vorgang des Nehmens dadurch sogar einfacher zu programmieren, schließlich muß eine Bedingung weniger überprüft werden, doch wollen wir es wie die Mehrheit der Adventureproduzenten halten und nur sichtbare Gegenstände auch greifbar sein lassen.

So wird denn auch die Holzhütte von unserer Hauptperson nur untersucht werden können, wenn sie sich in greifbarer Nähe befindet. Den Honig werden wir erst entdecken, nachdem wir die Korbflasche in unsere Hände genommen und geöffnet haben. Der Bär wird sich nur friedlich verhalten, wenn er mit dem Verzehr des Honigs anderweitig beschäftigt ist.

Alle diese Bedingungen müssen vor Ausführung eines Befehles zunächst auf ihre Richtigkeit überprüft werden. Bereits

unser kurzes Adventure mit nur sechs Räumen wird somit eine Vielzahl an Programmzeilen erfordern. Berücksichtigen wir nur die zu diesem Zeitpunkt der Entwicklung eingebrachten Verben untersuche, nimm, lege weg, öffne, benutze, zerstöre und unsere siebzehn Gegenstände, so müssen wir bereits Aktionen für 102 unterschiedliche Spielereingaben programmieren. Würden wir die Programmzeilen alle ohne besonderes Konzept hintereinanderschreiben, wäre es leicht, für die auftretenden, recht langen Ausführungszeiten des Programmes eine Erklärung zu finden. Daher werden wir den nun zu erstellenden Programmteil in mehrere Blöcke aufteilen.

Prinzipiell stehen uns zwei voneinander verschiedene Möglichkeiten zur Verfügung. Zum einen könnten wir für jeden Raum eine ausreichende Anzahl von Programmzeilen reservieren, und dort für jede mögliche Aktion des Spielers ein paar Statements reservieren. Diese Technik wird recht häufig gewählt, hat auch den Vorteil, daß sie wegen der direkten Ausführung eines Befehles, der ja zuvor nicht irgendwie aufbereitet werden muß, recht schnell ist, und daß vor allem ein recht übersichtliches Programm entsteht, welches bequem und einfach, ohne Nachsehen in irgendwelchen Listen, erweitert oder edititiert werden kann. Sollten Sie sich einmal mit dieser Ausführung eines Adventureprogrammes beschäftigen wollen, dann soll die folgende Realisation unseres Raumes 1 als Anregung genügen. Beachten Sie, daß es sich um ein Beispiel handelt, welches mit unserem Konzept nichts zu tun hat, überschreiben Sie daher bitte keine Zeilen unseres bisher erstellten Programmes.

```
100 INPUT"WAS SOLL ICH TUN";EINGABE$
200 REM EINGABE$ IN VERB$ UND OBJEKT$
ZERLEGEN (WIE BESPROCHEN)
300 ON RAUM GOTO 1000, 2000, 3000, 4000
5000, 6000
111 REM ABHAENGIG VON DER RAUMNUMMER
```

IN DEN BETREFFENDEN RAUM SPRINGEN

```
1000 PRINT CHR$(147)    REM RAUM 1
1010 PRINT"ICH BIN IM WALD".
1020 PRINT "ICH SEHE VIELE GROSSE BAE
UME".
1030:
1040 REM WEITERE GEGENSTAENDE DRUCKEN
1100 PRINT"ICH KANN NACH SUEDEN, OSTE
N."
1200 IF EINGABE$="S" THEN RAUM=6: GOTO 1
00
1210 IF EINGABE$="O" THEN RAUM=2: GOTO 1
00
1260 IF LEN(EINGABE$)<3 THEN PRINT "DAHI
N FUEHRT KEIN WEG !": GOTO 100
1300 IF EINGABE$="INV" OR "INVENTUR" THE
N GOTO 200
1400 IFVE$="UNT"ANDOB$="BAE"THEN PRINT
"ICH SEHE NICHTS BESONDERES":GOTO 100
1410 IFVE$="NIM"ANDOB$="BAE"THEN PRINT
"SO STARK BIN ICH NICHT.":GOTO 100
1999 PRINT "ICH VERSTEHE NICHT, WAS DU
MEINST.": GOTO 100
2000 ab hier Behandlung Raum 2
3000 ab hier Behandlung Raum 3
```

Wenn Sie sich obige Zeilen angesehen oder sogar in Ihren C64 eingegeben haben, werden Sie feststellen, daß auf eine noch einfachere Art ebenfalls ein funktionsfähiges Adventure entstehen kann, und Sie werden sich vielleicht fragen, warum wir nicht dieses Konzept verwendet haben, wo es zudem auch noch ohne lange Kommentare zu verstehen ist ?

Nun, einerseits werden besonders umfangreiche Adventures weitaus mehr Speicherplatz benötigen, denn wir würden es nicht vermeiden können, identische Routinen mehrmals

aufzuführen, da wir, um bei einem bekannten Beispiel zu bleiben, die Flasche in jedem Raum ablegen (sechs identische Programmzeilen) und, nicht zu vergessen, auch wieder nehmen können müssen. Diesen zwölf Zeilen stehen in unserem Adventuresystem ganze zwei Programmzeilen gegenüber. Allein das wäre schon Grund genug, doch auch das Argument der Übersichtlichkeit und Editierfreundlichkeit ist schnell zu entkräften.
Stellen sie sich vor, bei der Erweiterung des Adventures werden zur Fortführung der Handlung weitere Flaschen benötigt, so daß wir aus diesem oder irgendwelchen anderen Gründen aus der Flasche lieber ein kleines Holzfaß machen wollen. Hatte Ihr Adventure zu diesem Zeitpunkt bereits vierzig Räume, dürfen Sie nun in entsprechend vielen Zeilen aus ´Nimm Flasche´ ein ´Nimm Faß´ machen, entsprechendes gilt für ´Leg Flasche, Oeffne Flasche´ usw.
Haben Sie sich hingegen an unser Adventuresystem gehalten, ändern Sie Zeile 305, fertig !

Vergessen wir daher diese Methode und wenden wir uns wieder unserem Adventure zu. Wir lassen die Handlung ebenfalls in genau definierten Programmzeilen ablaufen, doch enthält nun jeder Block die Behandlung jeweils eines Verbes.

Unser Treiber hat bereits Verb- und Objektnummer ermittelt, daher können wir diese Blöcke gezielt anspringen. So ist es sinnvoll, allen Zeilen, die mit der Ausführung des Befehles ´untersuche´ beschäftigt sind, Programmzeilen ab 5000 zuzuteilen, die Aktion ´nim´ wird entsprechend ab Zeile 6000 realisiert.

Der Steuerung des Programmablaufes nimmt der Adventuretreiber anhand der Verbnummer in Zeile 2200 vor:

```
2180 REM          UNT    NIM    LEG    OEF
BEN   ZER
2200 ON VN GOTO 5000,6000,7000,8000,9000
,10000
```

Mit Programmzeile 5000 beginnt nun der Block ´UNTERSUCHE´. Es ist leicht einzusehen, daß uns gerade hier der größte Arbeitsaufwand abverlangt wird.
Nicht nur, daß auf *UNTERSUCHE HÜTTE* die Nachricht *IN DER HUETTE STEHT EIN ALTES REGAL.* ausgegeben und das Regal als sichtbares Objekt erscheinen muß. Nein, wir müssen auch sicherstellen, daß die Hütte an dem gerade besuchten Ort zur Verfügung steht, und gegebenenfalls eine Mitteilung wie *SO ETWAS SEHE ICH HIER NICHT.* an den Spieler übermitteln.

Diese Überlegungen zeigen bereits den Aufwand der Adventureprogrammierung, glücklicherweise lassen sich aber auch zahlreiche unterschiedliche Spielereingaben von ein und derselben Programmzeile beantworten.

So wird unser Programmteil, der die untersuchenden Handlungen des Spielers bearbeitet, zunächst prüfen, ob der Gegenstand, über den nähere Informationen gewünscht werden, entweder in Reichweite des Spielers, somit im gleichen Raum, oder gar in dessen Besitz ist, ansonsten kann sofort mit der Eingabe des nächsten Zuges begonnen werden:

```
5000 IF OB(N)<>SP AND OB(N)<>-1 THEN GOT
O 5900
5900 REM  GEGENSTAND NICHT VORHANDEN
5990 PRINT "SO ETWAS SEHE ICH HIER NICHT
 !" : GOTO 1080
```

DIE BEDINGUNGEN

Abgesehen von diesen zwei Bedingungen, lassen sich eine Reihe weiterer Voraussetzungen formulieren, ohne die ein Adventure nicht zu realisieren sein wird:

Objekt ist im Raum
Objekt wird vom Spieler mitgeführt
Objekt ist nicht im Raum
Flag ist gesetzt
Flag ist nicht gesetzt
Spieler ist in einem bestimmten Raum

Wie wir später noch sehen werden, erhebt diese Liste keinen Anspruch auf Vollständigkeit, denkbar ist beispielsweise eine Invertierung der letzten Bedingung, bestimmte Aktionen können in einem gewissen Raum nicht ausgeführt werden, wohl aber in jedem anderen Raum.

Einer Erläuterung bedürfen die an dieser Stelle aufgetauchten Flags. Es handelt sich hierbei um Signalschalter, die wir mittels eines weiteren Feldes realisieren wollen. Jedoch sollen diese Feldvariablen nur zwei Zustände annehmen können, entweder ist ein Flag gesetzt, dann soll der Inhalt der Variablen -1 sein, oder es ist nicht gesetzt, was einer Null entsprechen soll.
Diese Flags dienen unserem Adventure zur Identifizierung bestimmter Zustände:

ist eine Tür geöffnet oder verschlossen ?
wurde ein Hindernis beseitigt oder nicht ?
soll ein Ungeheuer dem Treiben des Spielers ein Ende setzen, oder darf er weiterziehen ?

In der Programmpraxis wird die Formulierung einer einzigen Bedingung meist jedoch nicht zur eindeutigen Klassifizierung der Situation ausreichen, so daß logische Verknüpfungen mehrerer Bedingungen erforderlich werden, um zu gewährleisten, daß einige Handlungen nicht völlig zusammenhanglos und zu einem falschen Zeitpunkt durchgeführt werden.

Bei diesen logischen Operationen wird es sich um UND- beziehungsweise ODER- Verknüpfungen handeln, für unsere Adventurepraxis heißt das, entweder müssen beide (AND) oder nur die eine oder die andere (OR) von zwei oder mehreren Verknüpfungen erfüllt sein. Können wir nicht auf mehr als zwei Bedingungen verzichten, wird meist auch eine Klammerung unumgänglich sein. Ein Beispiel soll dies deutlich machen:

> Denken wir an den Sprengstoff in unserer Kiste. Er soll dazu benutzt werden, allzu voreiligen Spielern die Freude am schnellen Weiterkommen zu vergällen.
> Eine Warnung ist vor der Explosion aus Fairneßgründen jedoch durchaus angebracht, weshalb auf *Untersuche Sprengstoff* die Meldung *Er sieht sehr explosiv aus* ausgegeben werden soll. Wenn der Spieler dann noch versucht, ihn zu nehmen, trägt er selbst die Schuld an seinem Ende.
>
> Hat der Spieler tatsächlich obige Eingabe gemacht, wird der Treiber die Nummern 1 (untersuche) und 9 (Sprengstoff) zur Verfügung stellen. Unser Problem liegt nun darin, daß wir uns nicht mit der Überprüfung eines Raumes zufrieden geben können, denn wir dürfen nicht vergessen, daß der Spieler zwar den Sprengstoff nicht greifen und mitnehmen kann, wohl aber die Holzkiste, und wenn er die Kiste mit sich führt, hat er auch den Sprengstoff bei sich. Prüfen wir daher, ob der Spieler den Sprengstoff gemeint hat, **UND** gleichzeitig die Holzkiste (Objekt Nummer 7) im gerade besuchten Raum **ODER** im Inventory des Abenteurers ist. Der eigentliche Handlungsgegenstand (Nummer 9) steht nicht zur Verfügung, nach Zeile 5000 wird der Programmablauf daher mit Zeile 5900 fortgestetzt werden.

Daher muß die Ausführung des Befehles *Untersuche Sprengstoff* folgendermaßen formuliert werden:

```
5904 IFN=9AND(OB(7)=SPOR OB(7)=-1)THENPR
INT"ER SIEHT SEHR EXPLOSIV AUS !":GOTO10
80
```

DIE AKTIONEN

Das vorangegangene Beispiel hat uns die Ausführung der wohl einfachsten Aktion, die Ausgabe einer Mitteilung an den Spieler, deutlich gemacht.
Meist ist dieser Print - Befehl jedoch nur Beiwerk zu anderen Handlungen, wie Manipulationen von OB(), um dem Abenteurer in einer Rückmeldung die Ausführung des Befehles zu bestätigen.
Befehle wie Nimm, Leg, Zerstöre und auch Öffne bedeuten für einen Gegenstand immer eine Ortsveränderung. Entweder wird er neuerdings vom Spieler mitgeführt, oder er gehört nun zum Inventar eines bestimmten Raumes, vielleicht verschwindet dieser Gegenstand auch ganz aus dem Spiel. Stellen Sie sich nur die Verblüffung des Spielers von Goldrausch vor, wenn der Bär den Honig offensichtlich genossen hat, die Flasche aber immer noch irgendwo zu finden ist !

Eine andere, ebenfalls häufig benötigte Aktion ist das Setzen oder Löschen der zuvor erwähnten Flags, um stufenweise die Voraussetzungen für spätere Handlungen zu schaffen.

Vergessen dürfen wir auch auf keinen Fall bedingte Ortswechsel des Spielers, welche durch Zauberei oder harmlose Aktionen ausgelöst werden. Eine spätere Erweiterung von Goldrausch könnte aus der Höhle des Bären ein riesiges Labyrinth von Gängen machen, und nach

Untersuche Höhle befindet sich der Spieler mitten in der Höhle, obwohl er diese nicht durch eine Bewegung in die entsprechende Himmelsrichtung betreten hat. Aber durch den erhöhten Programmieraufwand wird unser Adventure nur realistischer werden, denn wie soll man eine Höhle gründlich untersuchen, ohne sie zu betreten ?

Ähnlich wie bei den Bedingungen, können wir auch hier eine Liste möglicher Aktionen zusammenstellen, wobei allerdings wieder kein Anspruch auf Vollständigkeit erhoben wird. Jedoch sind sie ein ausreichender Grundstock, und es ist möglich, allein mit den vorgestellten Bedingungen und Aktionen gute Adventures zu schreiben.

Mitteilung an den Spieler ausgeben
Objekt verschwindet
Objekt kommt ins Inventory
Objekt erscheint neu im Raum
Flag wird gesetzt
Flag wird gelöscht
Spieler wird in anderen Raum versetzt

PROGRAMMIERUNG DER BEFEHLSAUSFÜHRUNG

Im folgenden finden Sie Hinweise zu Besonderheiten, die bei einzelnen Befehlen beachtet werden müssen. Dabei werde ich nur einige typische Programmzeilen erläutern, entnehmen Sie die restlichen Zeilen bitte dem Listing der Miniversion von Goldrausch am Ende dieses Kapitels.

AKTION: UNTERSUCHE

Zunächst steht fest, daß das Ergebnis der Untersuchung eines Gegenstandes immer in Form eines Satzes mitgeteilt wird.
Häufig wird es sich dabei um ein und denselben Satz handeln (Ich sehe nichts besonderes), weshalb wir die Mitteilungen an den Spieler in solchen Fällen nicht direkt ausgeben, sondern wir weisen den Text bei Programmstart einer Variablen zu und drucken deren Inhalt zu gegebener Zeit aus.
Da uns für eine Basiczeile nur 80 Zeichen zur Verfügung stehen, und wir in der betreffenden Zeile auch noch die Bedingungen formulieren müssen, wird es sich manchmal nicht vermeiden lassen, auch einige Meldungen vorzubereiten, an denen nur ein einziges Mal Bedarf besteht.

```
600 REM ------------------- MITTEILUNGEN
601 MS$(1)="ICH SEHE NICHTS BESONDERES."
602 MS$(2)="SO STARK BIN ICH NICHT."
603 MS$(3)="WIE STELLST DU DIR DAS VOR ?
"
604 MS$(4)="DER BAER NIMMT DEN HONIG UND
"
605 MS$(5)="VERSCHWINDET IN DER TIEFE DE
R HOEHLE."
```

Bereits erläutert wurde Zeile 5000, sie stellt für den gerade aktuellen Gegenstand durch Überprüfung des Wertes in OB() sicher, daß das Objekt sich in der Nähe des Spielers befindet.
Trifft dies zu, wird entsprechend der Objektnummer N die für diesen Gegenstand zuständige Programmzeile ermittelt. Sind alle dort programmierten Bedingungen erfüllt, wird der Rest der Zeile abgearbeitet und der Programmlauf anschließend mit dem Neuaufbau des Bildschirmbildes fortgesetzt:

```
5002 IF N=1 THEN PRINT MS$(1):GOTO1080
5003 IF N=3 THEN PRINT MS$(1):GOTO1080
5004 IF N=4 THEN PRINT"IN EINR ECKE STE
HT EIN REGAL.":OB(8)=SP:GOTO1080
```

Wird die Holzhütte untersucht (N=4), entdeckt der Spieler das Regal. Dieses war bislang in Raum 0, dem Lagerraum, und wird nun im Raum des Spielers positioniert, weshalb der Adventuretreiber beim anschließenden Durchlaufen der Zeilen 1170 bis 1210 unter anderem auch *ein klappriges Regal* ausgibt.

Nun ist es aber, wie im Fall des Regales, denkbar, daß ein Objekt auftaucht, welches der Spieler anschließend an sich nimmt.
Dann wäre eine Formulierung wie in 5004 nicht mehr ausreichend, da die Flasche würde bei jeder weiteren Untersuchung des Regales dem Inventory automatisch entnommen und wieder in den Raum gestellt werden.

Diesen Fehler können wir nur vermeiden, wenn wir uns zuvor vergewissern, daß die Flasche noch nicht im Spiel war, somit also noch immer im Lagerraum ist.

```
5008 IF N=8 AND OB(5)=OTHEN PRINT"AUF DE
```

```
M REGAL STEHT EINE KORBFLASCHE.":OB(5)=S
P:GOTO1080
5009 IF N=8 AND OB(5)<>O THEN PRINT MS$(
1):GOTO1080
```

Bei allen weiteren Untersuchungen des Regales wird der Spieler nichts besonderes mehr feststellen (Zeile 5009).

Ein weiteres Beispiel soll den Gebrauch der Flags zeigen. Hat der Spieler die Schatztruhe gefunden, ist diese zunächst mittels einer Eisenkette verschlossen. Nachdem er dieses Hindernis beseitigt hat, muß die Truhe selbstverständlich erst geöffnet werden, ehe der wertvolle Inhalt sichtbar wird. Um diese drei Zustände dem Spieler deutlich zu machen, wäre es möglich, weitere Gegenstände, wie eine mit einer Eisenkette verschlossene Truhe, eine unverschlossene Truhe, und eine geöffnete Truhe, einzuführen. Dieser recht hohe Programmieraufwand, der nebenbei natürlich auch den zur Verfügung stehenden Speicherplatz einschränkt, läßt sich durch den Einsatz von Signalschaltern umgehen. Sinnvollerweise fertigen Sie während der Konstruktion Ihres Adventurers eine Tabelle an, damit Sie auch wirklich die richtigen Flags umstellen:

Flag	0	-1
1	Kette heil	Kette zerstört
2	Truhe zu	geöffnet

Vielleicht werden Sie nun fragen, warum ausgerechnet -1 und nicht ein beliebiger anderer Wert, oder warum nicht nur ein einziges Flag mit verschiedenen Ziffern für jeden möglichen Zustand.

Der Grund ist, daß wir uns die Tatsache zunutze machen, daß für den Basic - Interpreter ein wahrer Ausdruck immer durch eine -1 repräsentiert wird, weshalb wir in den IF - Statements keine Vergleichswerte angeben müssen:

```
IF FL(1)=-1  entspricht  IF FL(1)
IF FL(1)=0   entspricht  IF NOT FL(1)
```

Für unser Beispiel gelten demnach folgende Zeilen:

```
5011 IFN=11ANDNOTFL(1) THENPRINT"SIE IST
 MIT EINER EISENKETTE VERSCHLOSSEN.":GOT
O1080
5012 IF N=11ANDFL(1)ANDNOTFL(2)THENPRINT
"VON AUSSEN SEHE ICH NICHTS BESONDERES."
:GOTO1080
5013 IF N=11ANDFL(1)ANDFL(2)THENPRINT"SI
E IST VOLLER SILBERMUENZEN.":OB(12)=SP:G
OTO1080
```

Ein weiterer Punkt, dem wir besondere Beachtung schenken müssen, findet seinen Grund in der Auslegung unserer Wortanalyse.
Zur Ausstattung der ersten zwei Räume gehören die völlig identischen Objekte 1 und 2. Wird die Eingabe des Spieles überprüft, so wird als Objektnummer immer nur eine 1 ermittelt werden können, welche bei einem Aufenthalt des Spielers in Raum 2 nicht den Tatsachen entspricht.
Um Eingabefehler richtig quittieren zu können, hatten wir jedoch Programmzeile 5000 eingesetzt, welche uns nun auch die Behandlung dieser Sonderfälle ermöglicht.
So lösen wir das Problem *Untersuche Baeume* für Raum 2 folgendermaßen:

```
5901 IF N=1 AND SP=2 THEN PRINTMS$(1):GO
TO1080
```

Gleiches gilt auch für den Bären. Solange wir die Wortlänge nicht auf vier erhöhen, wird unser Adventuretreiber den BAEren nicht von den BAEumen unterscheiden können:

```
5910 IFN=1ANDSP=5ANDOB(14)=5THENPRINT"IC
H SCHEINE SEINEN APPETIT ANZUREGEN !":GO
TO1080
```

Zum Abschluß der Routine müssen wir eine letzte Zeile vorsehen, die, wenn in keiner der vorangegangenen Programmzeilen alle Bedingungen erfüllt waren, ohne jede Bedingung durchgeführt wird. Hiermit werden wiederum nicht definierte Spieleingaben abgefangen, und wir verhindern, daß, wenn keine zutreffende Bedingung gefunden wurde, die Zeilen der anderen Verben abgearbeitet werden, und somit der Programmlauf außer Kontrolle gerät. Selbst wenn wir eine Handlung, die für den Ablauf des Adventures nicht notwendig ist, vergessen, wird ein Spieler das niemals bemerken:

```
5990 PRINT "SO ETWAS SEHE ICH HIER NICHT
 !" : GOTO 1080
```

AKTION: NIMM

Genau so, wie der Spieler einen Gegenstand nur untersuchen kann, der sich in seiner Nähe befindet, so wird er ihn auch nur nehmen können, wenn der Gegenstand entweder im gleichen Raum wie der Spieler oder bereits in dessen Taschen ist.

```
6000 IF OB(N)<>SP AND OB(N)<>-1 THEN GOT
O 6900
```

Danach wäre es an sich ganz einfach, den Befehl auszuführen, für eine Reihe von Objekten sollten wir dem Abentuerer seinen Wunsch jedoch versagen.
So sollten wir berücksichtigen, daß eine vollgefüllte Eisentruhe vermutlich zu gewichtig für einen einzelnen Menschen ist, ebenso können wir selbst dumme Versuche des Spielers wie *Nimm Baum* nicht unbeachtet lassen.

```
6001 IF N=1 THEN PRINT MS$(2):GOTO 1080
```

```
6005 IF N=11THEN PRINT MS$(2):GOTO 1080
```

Natürlich kann es auch noch schlimmer kommen, denn wir müssen dem Spieler selbstverständlich eine ausreichende Anzahl von Fallen stellen, damit ihm der Sieg nicht zu einfach gemacht wird. So sollte laut unserem Drehbuch eine Berührung des Sprengstoffes zur Explosion und damit zum Spielende führen. Gleiches gilt für den Fall, daß er versucht, sich der Nuggets zu bemächtigen, ohne dem Bären zuvor einen Leckerbissen angeboten zu haben. Ebenso fatal für den Spieler müßte sein Versuch sein, den Bären zu nehmen:

```
6015 IF N=1 AND SP=5 THEN MS$(0)="DER BA
ER HAT MICH ERSCHLAGEN.":GOTO4500
6018 IF N=17 AND NOT FL(3) THEN MS$(0)="
DER BAER STUERZT SICH AUF MICH.":GOTO450
0
6900 IF N=9THENMS$(0)="BEI DER BERUEHRUN
G IST DER SPRENGSTOFF EXPLODIERT.":GOTO
4500
```

Anmerkung: In Zeile 4500 beginnt eine Routine, welche bei nicht gewonnenem Spiel aufgerufen wird. MS$(0) ist in dieser Routine für eine erläuternde Nachricht vorgesehen.

Flag 3 wird gesetzt, wenn der Bär im Besitz des Honigs ist.

Steht dem Versuch des Spielers, sich an irgendwelchen Gegenständen unseres Adventures zu bereichern, nichts im Wege, müssen wir dem betreffenden Objekt nur die neue Platznummer, eine -1, zuweisen:

```
6010 IF N=5 THEN OB(5)=-1:PRINT"O.K.":GO
TO1080
```

AKTION: LEG

Irgendwann jedoch wird er sich dieser Objekte auch wieder entledigen wollen, entweder, weil wir als Programmierer seine Tragfähigkeit begrenzt haben, oder weil er sie irgendwie benutzen will.
In der Praxis wirft die Leg weg - Routine kaum Probleme auf, darüberhinaus überzeugt sie durch ihre Kürze.
Denn die einzige Voraussetzung ist, daß der wegzulegende Gegenstand im Besitz des Spielers ist, was für alle Objekte in Zeile 7000 getestet wird:

```
7000 IF OB(N)<>-1 THEN PRINT"SO ETWAS HA
BE ICH DOCH GAR NICHT.":GOTO1080
7900 OB(N)=SP:PRINT"O.K.":GOTO1080
```

Irrtümer des Spielenden werden sofort erkannt und entsprechend quittiert. War der Gegenstand im Inventory, wird ihm mit einer Änderung des Inhaltes von OB() ein neuer Aufenthaltsort zugewiesen.

Prinzipiell reichen diese beiden Zeilen aus, es kann jedoch erforderlich sein, neben dem Positionswechsel weitere Reaktionen durchzuführen. So hatten wir für Goldrausch vorgesehen, daß der Bär sich die Flasche greift und damit in den Tiefen der Höhle verschwindet.
Diese Aktion darf natürlich nur durchgeführt werden, wenn der Spieler sich gerade in Raum 5 befindet, an allen anderen Orten wird der Vorgang ebenfalls durch Zeile 7900 ausgeführt.

```
7020 IFN=5ANDSP=5 THEN OB(5)=0:FL(3)=-1:
PRINTMS$(4):PRINTMS$(5):OB(14)=0:GOTO108
0
```

AKTION: OEFFNE

Gleich zu Beginn dieses Programmteiles steht der nun schon bekannte Test, um technische Fehler zu vermeiden. Wiederum ist es nötig, sowohl den Raum als auch das Inventory zu überprüfen, schließlich wird der Spieler eine zu öffnende Tür nicht erst nehmen müssen.

```
8000 IF OB(N)<>SP AND OB(N)<>-1 THEN PRI
NT"SO ETWAS IST HIER NICHT.":GOTO1080
```

Die Aktion selbst wird meist aus dem Setzen eines Flags und der Ausgabe einer Mitteilung bestehen:

```
8025 IF N=11 AND FL(1) THEN PRINT"O.K. -
 DER DECKEL KLAPPT NACH HINTEN.":FL(2)=-
1:GOTO1080
```

Nicht vergessen dürfen wir unwichtige Aktionen, die aber von der Lage der Dinge her möglich sind. Als Beispiel mag wieder unsere Flasche dienen. Ein logisch denkender Abenteurer wird sie vor einer näheren Untersuchung erst öffnen wollen und wäre sicher überrascht, wenn ihm dies nicht gelingt bloß weil es für das Spiel unwichtig war.

```
8010 IF N=5 THEN PRINT "O.K.":GOTO1080
```

Den Abschluß dieses Blockes gewährleistet, daß neben den technischen Falscheingaben auch inhaltliche Unmöglichkeiten wie *Öffne Honig* erkannt werden.

```
8999 PRINT"ICH VERSTEHE NICHT, WAS DU ME
INST.":GOTO1080
```

ZUSAMMENFASSUNG

Ich hoffe, daß Ihnen mit den Erläuterungen zu diesen wohl häufigsten Handlungen die Ausführung der eigentlichen Spielprogrammierung klar geworden ist. Prinzipiell läßt sich zusammenfassend sagen, daß die einzelnen Aktionen durch zwei Programmzeilen umklammert werden.

So wird zu Beginn eines Blockes getestet, ob von der technischen Seite her die Durchführung der Aktion möglich wird, abschließend wird sichergestellt, das der Programmlauf selbst bei Auftreten irgendeines Fehlers korrekt fortgesetzt wird.

Das Erkennen der richtigen Voraussetzungen für die Durchführung eines Befehles ist dabei, ebenso wie die Aktion selbst, meist eine recht einfach zu programmierende Angelegenheit, die bei unserem Adventuresystem zudem noch standardisiert wurde. Folgende Zusammenstellung soll Ihnen bei der Realisation Ihrer ersten eigenen Adventures eine Hilfe sein:

BEDINGUNG	*IM BASICPROGRAMM*
Objekt ist im Raum	OB(objekt)=SP
Objekt wird vom Spieler mitgeführt	OB(objekt)=-1
Objekt ist nicht im Raum	OB(objekt)<>SP
Flag ist gesetzt	FL(x)=-1
Flag ist nicht gesetzt	FL(x)=0
Spieler ist in einem bestimmten Raum	SP=Raumnummer

AKTIONEN	*IM BASICPROGRAMM*
Mitteilung an den Spieler ausgeben	PRINT"" oder MS$
Objekt verschwindet	OB(Objekt)=0
Objekt kommt ins Inventory	OB(Objekt)=-1

BEDINGUNG	IM BASICPROGRAMM
Objekt erscheint neu im Raum	OB(Objekt)=SP
Flag wird gesetzt	FL(x)=-1
Flag wird gelöscht	FL(x)=O
Spieler wird in anderen Raum versetzt	SP=Raumnummer

ERLÄUTERUNG DER WICHTIGSTEN VARIABLEN:

AR	Anzahl der Räume eines Adventures
AO	Anzahl der Objekte
AV	Anzahl der Verben
DU	Auswege aus einem Raum
I	Zählvariable in Schleifen
N	Nummer des benutzten Objektes
OB	Aufenthaltsort eines Objektes
RI	Zählvariable für Richtungen
SP	Aufenthaltsraum des Spielers
VN	Nummer des benutzten Verbes
WL	signifikante Wortlänge
EI$	Eingabe des Spielers
EO$	Gegenstand der Spielereingabe
EV$	Verb der Spielereingabe
MS$	auszugebende Mitteilungen
OB$	Beschreibung eines Objektes
RN$	Rufname eines Objektes
VE$	Verb

DIE LETZTEN SCHRITTE

Bevor unser Adventure den in Kapitel 2 entwickelten Vorstellungen entspricht, müssen wir uns noch Gedanken zu drei weiteren Programmteilen machen.
So ist es für einen einwandfreien Spielablauf unbedingt erforderlich, dem Spieler die Möglichkeit zu einer schnellen Inventur in die Hand zu geben.

Es wäre duchaus möglich, diese Routine auf gleiche Art und Weise wie die Behandlung der Verben auszuführen. Da es sich hierbei jedoch um eine grundsätzliche Funktion der Adventures handelt, werden wir unseren Treiber in geeigneter Weise ergänzen.

INVENTUR

Inventur gehört, wie auch die später zu erstellenden Routinen zum Abspeichern und Laden des Spielstandes, zur Befehlsgruppe der sogenannten Ein - Wort - Befehle. Für diesen Programmteil haben wir zwischen den Programmzeilen 1470 (Ende der Bewegungsroutine) und 2000 (Analyse der Eingabe) ausreichend Platz gelassen.

Um unseren Treiber nicht unnötig lange damit zu beschäftigen, einen jeden Spielerzug mit den Befehlen dieser Sondergruppe zu vergleichen, entscheiden wir zunächst, ob es sich um eine dieser speziellen Anweisungen oder um eine reguläre Eingabe handelt.
Die übliche Spielereingabe wird eine Länge von mindestens sieben Buchstaben haben, gehen wir daher davon aus, daß es sich, wenn die Länge der Eingabe geringer ist, um einen Ein - Wort Befehl handeln:

```
1490 IF LEN(EINGABE$)>6 THEN GOTO 2000
```

Handelte es sich bei der Eingabe des Spielers nicht um eine reguläre Verb/Objekt - Kombination, werden alle Zeilen, in denen die Behandlung eines EW - Befehles beginnt, der Reihe nach durchlaufen, bis eine Übereinstimmung der ersten drei Buchstaben festgestellt wird.

In der Praxis besteht die Ausführung der Inventur im Durchlaufen einer einfachen Schleife, welche alle Gegenstände des Adventures, die mit -1 gekennzeichnet sind, auf dem Bildschirm ausdruckt:

```
1499 REM ------- START INVENTUR --------
1500 IF LEFT$(EINGABE$,3)<>"INV" THEN GO
TO 2000
1510 PRINT"ICH TRAGE FOLGENDES MIT MIR:"
1520 FOR I=1 TO AO
1530 IF OB(I)=-1 THEN PRINT OB$(I)
1540 NEXT I
1550 GOTO 1080
1560 REM -------- ENDE INVENTUR --------
```

DIE TITEL

Rein technisch gesehen haben wir unser Programm mit diesen Zeilen fertiggestellt. Alle gewünschten Funktionen stehen zur Verfügung und einem Spiel steht nichts mehr im Wege. Doch wie lange soll der Abenteurer in unserer Welt umherirren, wann hat er sein Ziel erreicht ?

Handlungsmäßig ist die Miniversion von Goldrausch beendet, wenn sowohl die Nuggets als auch die Silberstücke sich im Besitz des Spielers befinden.

Eine einzige Basic - Zeile reicht aus, um in diesem Falle alle weiteren Eingaben zu unterbinden und einen Programmteil anzuspringen, welcher dem Spieler in

angemessener Form seinen Erfolg mitteilt und das Spiel beendet.

```
1340 IF OB(12)=-1 AND OB(17)=-1 THEN GOT
O 4800
```

Für Adventurespiele typisch ist jedoch ein völlig anderes, weniger ruhmreiches Ende. Da auch Spieler unserer Programme meistens gezwungenermaßen ihr Spiel etwas früher beenden werden, programmieren wir ein weiteres Endbild ab Zeile 4500.
Bei der Ausführung der spielbeendenden Handlungen hatten wir ja bereits eine entsprechende Nachricht vorbereitet, so daß der Spieler zumindest nicht über die Ursache seines Scheiterns im Unklaren gelassen werden muß.

```
4500 PRINT CHR$(147) REM SPIELER TOD ---
4600 PRINT "AUCH DAS NOCH !":PRINT:PRINT
MS$(0)
4610 PRINT"ICH BIN TOD !":PRINT
4620 INPUT"SOLL ICH ES NOCH EINMAL VERSU
CHEN";A$
4630 IF LEFT$(A$,1)="J"THEN GOTO 700
4640 PRINT CHR$(147):END
```

Dabei erlaubt es uns die Einbeziehung der individuellen Nachricht in MS$(0), dieses Titelbild für jede Situation und für jedes Adventure zu verwenden. Änderungen werden, sofern sie überhaupt erforderlich sind, nur wenig Aufwand verursachen. Aus diesem Grunde können obige Zeilen, ebenso wie die Siegesnachricht ab 4800, als weitere Ergänzungen unseres Adventuretreibers betrachtet werden.

Bedenken wir weiterhin, daß es für Adventures durchaus nicht typisch ist, sie zu laden und in einigen wenigen Stunden durchzuspielen, und programmieren wir zu guter Letzt noch ein reguläres Programmende.

Auch wenn sich solche Software auf dem Markt befindet, es zeugt nicht von der Benutzerfreundlichkeit eines Programmes, wenn es nur durch Drücken der Break- oder Resettaste verlassen werden kann.

Fügen wir daher einen weiteren Ein - Wort Befehl in unser System ein:

```
1500 IF LEFT$(EINGABE$,3)<>"INV" THEN GO
TO 1950
1950 IF LEFT$(EINGABE$,3)<>"END" THEN GO
TO 2000
1960 PRINT "DER AUTOR WUENSCHT IHNEN MEH
R GLUECK    FUERS NAECHSTE MAL !":END
```

Sinngemäß waren das die letzten Zeilen unseres Adventures, alle denkbaren Versionen des Spielendes haben wir realisiert. Doch wie nimmt sich dagegen der Beginn unseres Werkes aus ?

Starten wir unser Programm so wie es ist, besteht dessen erste Handlung darin, die Arbeitsvariablen vorzubereiten, eine Arbeit, die besonders bei größeren Adventures den Gedanken an einen Absturz des Programmes aufkommen läßt. Anschließend befindet man sich mitten im Programm, und ein unerfahrener Adventureneuling sieht den Sinn des Spieles vielleicht darin, eine Seight Seeing Tour durch eine elektronisch simulierte Welt anzustellen.

Schenken wir daher einer alten Volksweisheit Glauben, die besagt, daß der erste Eindruck oft entscheidend ist.
Machen wir uns die Mühe, unser Programm, bevor wir es als fertig bezeichnen, mit einem, oder besser noch, mit mehreren Titeln zu versehen.

Dem ersten Titel wird dabei die Aufgabe zufallen, in knapper, aber ansprechender Form, Aufklärung über die im

Speicher befindlichen Bytes zu geben, Programmtitel und -gattung zu nennen sowie Auskunft über den Autoren zu geben.

Das nächste Bild bringen wir auf den Schirm, bevor die Variablen initialisiert werden. Während dieses Zeitraumes kann dieser Titel Auskunft über die dem Spieler gestellte Aufgabe geben, und ihn in die Handlung einführen.

Gehört das Herausfinden der gestellten Aufgabe allerdings mit zur Spielidee, wird diese Tafel fehlen, und an den Programmtitel schließt sich eine Erklärung der Spielidee an, die sonst als drittes Titelbild implementiert werden muß.

Denn wie war es, als Sie Ihr erstes Adventure in die Hand und in den Speicher bekamen, kannten Sie Sinn und Ziel dieser Programme, oder saßen Sie rat- und tatenlos vor Ihrem Computer ?

Vergessen wir doch nicht, daß es immer wieder Anfänger geben wird. Anfänger, die froh sind, wenn sie nach dem Einschalten des Rechners ein Programm ohne Fehlermeldungen laden können. Eine grundsätzliche Spielanweisung bedeutet doch nur geringen zusätzlichen Aufwand, machen wir uns daher ans Werk und rechtfertigen ein mögliches Fehlen nicht

```
   Herzlich Willkommen zur Miniversion von
                  GOLDRAUSCH
 ________________________________________________
 Auf der Suche nach dem Glueck in der
 neuen Welt begegneten Sie vor einigen
 Tagen einem alten todkranken Mann, dem
 Sie in seinen letzten Stunden Beistand
 leisteten.
 Aus Dankbarkeit berichtete er Ihnen von
 seiner Goldmine und den dort versteckten
 Resten seines Vermoegens.

 Zahllosen Gefahren widerstanden Sie
 auf dem Weg dorthin; bald werden Sie Ihr
 Ziel erreicht haben und es wird sich
 zeigen, ob der Alte die Wahrheit ge-
 sprochen oder im Fiebertraum geredet
 hatte.
 ________________________________________________
       Wuenschen Sie Ratschlaege fuer
          Ihr weiteres Vorgehen ? j
```

etwa damit, daß das Herausfinden der Spielregeln bereits das erste beabsichtigte Hindernis wäre.

```
   C64 - Adventure System, Version 1.0
     (c) 1984   by Walkowiak
Stellen Sie sich einen Roboter vor, den
Sie mit zahlreichen Kommandos steuern
koennen. Ich bin dieser Roboter, und
ich werde mich fuer Sie den Gefahren
der verwegensten Abenteuer aussetzen.

Damit Sie mich sinnvoll agieren lassen
koennen, werde ich Ihnen die Situation,i
n der ich mich gerade befinde, jeweils g
enau beschreiben. Anschliessend sagen
Sie mir mit zwei Worten, wie zum Bei-
spiel UNTERSUCHE TUER, NIMM MESSER, was
ich tun soll.

Darueber hinaus verstehe ich die Be-
fehle INVentur    SAVE    LOAD
       VOKabeln   HELP    ENDE
       SCOre und INStruktionen

           (Taste druecken)
```

Zwei Beispiele haben Ihnen gezeigt, wie es gemeint war. In Ihren eigenen Adventurespielen stehen für diese Aufgaben die Programmzeilen 10 - 100, sowie 700 - 900 zur Verfügung.

Sinnvollerweise bauen Sie die allgemeinen Erläuterungen dabei als Unterprogramm ein, welches durch Return abgeschlossen wird.
Dies ermöglicht einen weiteren Ausbau des Adventuretreibers, und erlaubt es dem Spieler, sich die Regeln ohne eine Unterbrechung des Spieles in die Erinnerung zurückrufen zu können.

HINWEISE ZUM FOLGENDEN LISTING

Nachdem Sie die folgenden Zeilen in Ihren Commodore eingegeben haben, steht Ihnen die lauffähige Miniversion von Goldrausch zur Verfügung. Es gilt jedoch zu beachten, daß es sich nur um zusätzliche Programmzeilen handelt, alle im vorangegangenen Text erläuterten Zeilen müssen sich bereits im Speicher befinden.

Sie werden sehen, daß mit den bisher vorgestellten Techniken bereits anspruchsvolle Adventures entwickelt werden können, die den Vergleich mit entsprechender käuflicher Software nicht zu scheuen brauchen.

Es liegt nun an Ihnen, ob Sie das Buch zunächst beiseite legen und ein eigenes Abenteuerspiel entwickeln oder lieber weitere Features kennenlernen wollen, die aus einem guten ein perfektes Adventure machen können.

```
1 REM -- GOLDRAUSCH, MINIVERSION 1.0 --
2 REM        (C) 1984 BY WALKOWIAK
10 REM -- TITELBILD ZUM ADVENTURE
11 POKE53281,11:POKE53280,12:PRINTCHR$(1
42):REM ------------------------------ FARBEN
12 PRINT"
13 PRINT"
14 PRINT"
15 PRINT"
16 PRINT"
17 PRINT"
18 PRINT"       (C) 1984
19 PRINT"             BY
20 PRINT"       JOERG WALKOWIAK
21 PRINT"
22 PRINT"
23 PRINT"
24 PRINT"
25 PRINT"
26 PRINT"
27 PRINT"
28 PRINT"
29 PRINT"
```

```
30 PRINT"          |          |/
/  ▚▙__|
31 PRINT"          |__________|/ /
32 PRINT"                      /__________/
33 PRINT" ▐   AUS DEM DATA BECKER BUC
H
34 PRINT"▚▚       ADVENTURES-▛ /
35 PRINT"▚▚     UND WIE MAN SIE PROGRAMMI
ERT
36 PRINT"▛                /___________/";
99 FOR I=1 TO 10000:NEXT
100 REM ----------------------- KENNDATEN
101 :
120 AO=18
140 WL=3: REM --------------- WORTLAENGE
150 SPIELER=1
199 :
209 :
318 DATA"EINE EISENSTANGE","EIS",0
499 :
599 :
698:
699 REM ----------- 2, TITEL: EINLEITUNG
700 PRINT"□":POKE53280,0:POKE53281,0:PRI
NT"▚";CHR$(14);
710 PRINT"IERZLICH oILLKOMMEN ZUR \INIVE
RSION VON"
715 PRINTSPC(14)"▚▛|ΓL¯♣ ♠¯|▛"
720 PRINT"▐──────────────────────────────
──────"
725 PRINT"▐♣UF DER ♠UCHE NACH DEM ILUECK
 IN DER"
730 PRINT"NEUEN oELT BEGEGNETEN ♠IE VOR
EINIGEN"
735 PRINT"IAGEN EINEM ALTEN TODKRANKEN \
ANN, DEM"
740 PRINT"♠IE IN SEINEN LETZTEN ♠TUNDEN
IEISTAND"
745 PRINT"LEISTETEN."
750 PRINT"♣US ¯ANKBARKEIT BERICHTETE ER
⌐HNEN VON SEINER IOLDMINE UND DEN ";
```

```
755 PRINT"DORT VERSTECKTEN";:PRINT"-ESTE
N SEINES >ERMOEGENS."
760 PRINT"※♠AHLLOSEN |EFAHREN WIDERSTAND
EN ♠IE"
765 PRINT"AUF DEM oEG DORTHIN; BALD WERD
EN ♠IE ¬HR";
770 PRINT"♠IEL ERREICHT HABEN, UND ES WI
RD SICH"
775 PRINT"ZEIGEN, OB DER ♠LTE DIE oAHRHE
IT GE-"
780 PRINT"SPROCHEN, ODER IM -IEBERTRAUM
GEREDET"
785 PRINT"HATTE."
790 PRINT"------------------------------
----------"
799 :
835 :
855 :
875 :
880 PRINT"※      oUENSCHEN ♠IE -ATSCHLAEG
E FUER"
885 INPUT"             ¬HR WEITERES >ORGEHEN
 ";EI$
890 IF EI$="J" THEN GOSUB 900
895 GOTO 1000
899 :
900 REM -------- 3. TITEL: INSTRUKTIONEN
910 PRINT"♫";CHR$(14)
920 PRINT"  -64 - ♠DVENTURE ♠YSTEM, >ERS
ION 1.0"
930 PRINT"      (C) 1984  BY oALKOWIAK
"
940 PRINT"------------------------------
----------"
950 PRINT"♠TELLEN ♠IE SICH EINEN -OBOTER
 VOR, DEN"
951 PRINT"♠IE MIT ZAHLREICHEN ⌐OMMANDOS
STEUERN"
952 PRINT"KOENNEN. ¬CH BIN DIESER -OBOTE
R, UND"
953 PRINT"ICH WERDE MICH FUER ♠IE DEN |E
```

```
FAHREN"
954 PRINT"DER VERWEGENSTEN ♣BENTEUER AUS
SETZEN."
955 PRINT"▣▔AMIT ♠IE MICH SINNVOLL AGIER
EN LASSEN"
956 PRINT"KOENNEN, WERDE ICH ╮HNEN DIE ♠
ITUATION"
957 PRINT"IN DER ICH MICH GERADE BEFINDE
, JEWEILS GENAU BESCHREIBEN.";
958 PRINT" ♣NSCHLIESSEND SAGEN":PRINT"♠I
E MIR MIT ZWEI ○ORTEN, WIE ZUM │EI-"
959 PRINT"SPIEL ╱╱│▔▁♠╱─│▔ │╱▔─, ╱╲╲╲ ╲▔
, WAS"
960 PRINT"ICH TUN SOLL."
961 PRINT"▣▔ARUEBERHINAUS VERSTEHE ICH D
IE │E-"
962 PRINT"FEHLE ╮╱╳▔╱│ ╱▁ UND ▔╱▔▔."
975 PRINT"────────────────────────────────
────────"
980 PRINT"▣▣"SPC(11)"(│ASTE DRUECKEN)";
985 GET EI$: IF EI$="" THEN 985
990 PRINT"▢":PRINTCHR$(142):RETURN
995 :
999 REM ------- BEGINN ADVENTURE-DRIVER
1000 PRINT"▢":PRINTCHR$(142)
1070 PRINT"▢":POKE 53280,0:POKE 53281,0:
PRINT"▤"
1080 PRINT"▢" : REM BEGINN NEUER ZUG --
1240 PRINT"▢ICH KANN NACH ";
1260 IF DURCHGANG(SPI,RICHTUNG)=0 THEN G
OTO 1310
1330 PRINT"▤──────────────────────────────
────────"
1390 POKE 211,0:POKE 214,24:SYS 58732:PR
INT"▤";:INPUT"WAS SOLL ICH TUN";EI$:PRIN
T"▣▣";
1498 :
1561 :
1899 REM ---------- INSTRUKTIONEN / ENDE
1900 IF LEFT$(EINGABE$,3)<>"INS" THEN GO
TO 1950
```

```
1910 GOSUB 900
1920 GOTO 1080
1950 IF LEFT$(EINGABE$,3)<>"END" THEN GO
TO 2000
1960 PRINT"□DER AUTHOR WUENSCHT IHNEN FU
ERS NAECHSTE■MAL MEHR ERFOLG !■■■■■":END
1990 :
2080 EO$=LEFT$(EO$,WL)
2169 :
2170 REM -- SPRUNGZIELE FUER AUSFUEHRUNG
2201 :
4498 :
4499 REM ------------------------- SPIELENDE
4610 PRINT"■■■ICH BIN TOD !":PRINT
4641 :
4799 :
4800 PRINT"□":REM ------ SPIELER SIEGT -
4810 PRINT"■HERZLICHEN GLUECKWUNSCH"
4820 PRINT"■■SIE HABEN IHRE AUFGABE GELO
EST"
4830 PRINT"■UND DUERFEN SICH AN EINEM AN
DEREN
4840 PRINT"■ADVENTURE VERSUCHEN."
4899 END
4998 :
4999 REM ---------- SPIELERZUG AUSFUEHREN
5005 IF N=5 THEN PRINT "DIE FLASCHE IST
GEFUELLT MIT HONIG.":GOTO1080
5006 IF N=6 THEN PRINTMS$(1):GOTO1080
5007 IF N=7 ANDOB(9)=0THENPRINT"IN DER K
ISTE LIEGT SPRENGSTOFF.":GOTO1080
5010 IF N=10THEN PRINT"IN DEM ERDLOCH LI
EGT EINE EISENTRUHE.":OB(11)=SP:GOTO1080
5014 IF N=12 THEN PRINT"■GENAU DAS SUCHE
ICH !!":GOTO1080
5015 IFN=13THENPRINT"ICH HABE EINEN BAER
EN AUFGESCHRECKT.":OB(14)=SP:GOTO1080
5017 IF N=15 THENPRINT"ZWISCHEN DEN BUES
CHEN IST EIN ERDLOCH.":OB(10)=SP:GOTO108
0
5018 IF N=16 THEN PRINT"SIE SEHEN SEHR S
```

```
TABIL AUS.":GOTO1080
5019 IF N=17 THEN PRINT"ES HANDELT SICH
UM REINES GOLD !":GOTO1080
5902 IFN=6 AND OB(5)=-1 THEN PRINT"ER IS
T SUESS UND GUT.":GOTO1080
5903 IF N=6 AND OB(5)<>-1 THEN PRINT"ICH
 HABE KEINEN HONIG !":GOTO1080
5905 IF M=9 THEN PRINT"DER SPRENGSTOFF S
IEHT BEDROHLICH AUS !":GOTO1080
6002 IF N=3 THEN PRINT MS$(2):GOTO 1080
6003 IF N=4 THEN PRINT MS$(3):GOTO 1080
6004 IF N=8 THEN PRINT MS$(2):GOTO 1080
6006 IF N=10THEN PRINT MS$(3):GOTO 1080
6007 IF N=13THEN PRINT MS$(3):GOTO 1080
6008 IF N=15THEN PRINT MS$(2):GOTO 1080
6011 IF N=6 THEN OB(5)=-1:PRINT"O.K.":GO
TO1080
6012 IF N=7 THEN OB(7)=-1:PRINT"O.K.":GO
TO1080
6014 IF N=12 THEN PRINT"O.K.":OB(12)=-1:
GOTO1080
6016 IF N=16 THEN PRINT"O.K.":OB(18)=-1:
GOTO1080
6017 IF N=17 AND FL(3) THEN PRINT"O.K.":
OB(17)=-1:GOTO1080
7010 IF N=6ANDSP=5THEN OB(6)=0:FL(3)=-1:
PRINTMS$(4):PRINTMS$(5):OB(14)=0:GOTO108
0
8005 IF N=4 AND SP=3 THEN PRINT"DIE HUET
TE WAR BEREITS OFFEN.":GOTO1080
8020 IF N=11 AND NOT FL(1) THEN PRINT"DA
S LAESST DIE KETTE NICHT ZU.":GOTO1080
9000 REM     VERB= BENUTZE
9010 IF N=16 AND SP=4 THEN PRINT"DIE KET
TE ZERSPRINGT.":FL(1)=-1:GOTO1080
9999 PRINT"ICH VERSTEHE NICHT, WAS DU ME
INST.":GOTO1080
10000 IF N=19 AND SP=4 AND OB(18)=-1 THE
N PRINT"DIE KETTE ZERSPRINGT.":FL(1)=-1:
GOTO1080
10010 IF N=19 AND SP=4 AND OB(18)<>-1 TH
EN PRINT"WOMIT ?":FL(1)=-1:GOTO1080
10999 PRINT"ICH VERSTEHE NICHT, WAS DU M
EINST.":GOTO1080
```

4. KAPITEL

- PERFEKTIONIERUNG -

VOM GUTEN ZUM PERFEKTEN ADVENTURE

Wenn Sie mit der Miniversion von Goldrausch gespielt oder selber ein Adventure auf der Basis der bislang erarbeiteten Techniken erstellt haben, werden Sie mit mir übereinstimmen, daß der Standard gängiger Programme der Gattung Adventures durchaus erreicht wurde.

Dennoch lassen sich ab und an Abenteuerspiele finden, die mit weiteren Extras aufwarten. Um sich über die Konkurrenz erheben zu können, wurde entweder ihre Benutzerfreundlichkeit erhöht, oder es fanden bei der Realisierung des Spieles auch die kleinen Nebensächlichkeiten des täglichen Lebens Verwendung, welche dem Spieler das Leben nun zusätzlich schwer machen.
Darüber hinaus existieren bereits einige Programme, die mit entsprechendem Werbeaufwand auf ihre noch nie dagewesenen Besonderheiten aufmerksam machen.
So wird auch dem Gehör des Spielers etwas geboten, von einzelnen Beeps und Pieps, die einen verstandenen oder aber einen undurchführbaren Befehl signalisieren über die akustische Untermalung sichtbarer Objekte bis hin zur Nachahmung der menschlichen Stimme. Leider ist der spielerische Nutzen gleich Null, und man sollte es sich mehr als zweimal überlegen, ob man sein Geld für speicherfüllende Effekthascherei opfert oder lieber eine um den gesparten Speicherplatz vergrößerte Adventurewelt vorzieht.

BENUTZERFREUNDLICHKEIT

wird für Geschäftsprogramme immer wieder gefordert, warum nicht auch für Spiele ?

Da gibt es doch tatsächlich Adventures, die dem Spieler nicht einmal mitteilen, auf welchen Wegen er einen gefährlichen Ort wieder verlassen kann. Sie warten nur darauf, ihn durch die nach Eingabe der verschiedensten Himmelsrichtungen auftauchende Mitteilung ´In diese Richtung kannst du nicht. - You can´t go in that direction.´ zur Verzweiflung zu treiben.

Gut, daß der Spieler sich die Haare raufen darf und soll, damit bin ich einverstanden, doch scheinen mir diese Programme eher dazu geschrieben worden zu sein, um mit dem Werbespruch ´Wir garantieren Ihnen, daß Sie Monate brauchen, um alle Orte dieses Adventures kennenzulernen !´ an den Adventurefan gebracht zu werden.

Diese Kritik kann unser Adventuresystem glücklicherweise nicht treffen, dennoch ist mit dem Fehlen jeder Möglichkeit zum Abspeichern des augenblicklichen Spielstandes eine Voraussetzung gegeben, die Freude an einem Spiel rasch zu vergällen.
Unglücklicherweise gilt das umso mehr für Adventures, bei deren Entwicklung wir einen besonderen Aufwand getrieben haben, und auf deren Komplexität und intelligent gestellte Fallen wir besonders stolz sein dürfen.

Helfen wir also dem Spieler, der sowieso mehr als nur sieben Leben haben muß, um alle möglichen adventuregemäßen Arten des Ablebens kennenlernen zu können, und schaffen wir die technischen Voraussetzungen dazu, daß er das Spiel, um eine Erfahrung reicher, ab dieser Stelle fortsetzen kann. Wenn er dann nicht rechtzeitig Gebrauch von dieser sinnvollen Erweiterung macht, kann er sich nur über sich selber ärgern.

Außerdem verhindern wir mit diesen Routinen einen Schnitt ins eigene Fleisch, denn während der Testphase eines Adventures wird sich immer wieder die Notwendigkeit kleiner

Programmänderungen zeigen, welche leider auch die Inhalte aller Variablen zunichte machen.
Was bleibt, ist nur ein neuer Start mit Run und die Wiederholung aller notwendigen Eingaben, um wieder in den Raum zu gelangen, in dem der Fehler aufgetreten war, um dann die Feststellung zu machen, daß die neue Version keineswegs eine Verbesserung bedeutet.

SAVE GAME / LOAD GAME

Überlegen wir zunächst, wodurch sich eine bestimmte Szene des Spieles von der Ausgangsstellung unterscheidet.

Sofort fällt uns der Spieler ein, der sich mehr oder weniger zielstrebig umherbewegt hat, ebenso denken wir an die Gegenstände, die er genommen oder abgelegt, zerstört, verbraucht oder gegessen hat, an Gegenstände, die nicht mehr existieren, ebenso wie an Dinge, die neu im Spiel aufgetaucht sind.

Neben diesen sichtbaren, und deshalb vielleicht weniger wichtigen, weil kontrollierbaren Variableninhalten, müssen wir ebenfalls zur Steuerung des Spielverlaufes gemachte Veränderungen retten, sonst kann es geschehen, daß eine Aktion trotz aller erforderlichen Gegenstände nicht den gewünschten Erfolg bringt.

Denken wir nur an die Flags, die nicht immer nur für den Verlauf einer zusammenhängenden Aktion, sondern auch für die Kontrolle scheinbar zusammenhangloser Ereignisse benutzt werden können.
Gleiches gilt für unsere Richtungstabelle, auch hier sind, wie wir später noch sehen werden, umfangreiche Veränderungen durchaus üblich.

Kurz gesagt: die Inhalte der Variablen SP, OB(), FL() und DU(,) müssen vor einem Spielabbruch als sequentielle Datei auf der Diskette bzw. Kassette gespeichert werden.

Exkurs: Externe Datenspeicherung

Neben der internen Datenspeicherung, bei der die Daten im eigentlichen Arbeitsspeicher der CPU abgelegt sind und ihr ständig zur Verfügung stehen, sind für den Umgang mit Daten, gleich welcher Art, auch externe Speicher notwendig, die den Speicher eines Computers fast beliebig erweitern und im wesentlichen der Archivierung und der Bereitstellung größerer Datenmengen dienen. Als Vertreter dieser Gattung sind beispielsweise Lochkarten und -streifen, Magnetbänder, Platten- und Walzenlaufwerke wie auch unsere 1541-Floppy oder die Datasette zu nennen.

Abgesehen von verschiedenen Kapazitäten und Zugriffszeiten, lassen sich auch Unterschiede in der Art der Datenspeicherung erkennen.

Speichergeräte, wie ein Lochstreifenstanzer/leser oder wie unsere Datasette, arbeiten dabei mit sogenannten sequentiellen Dateien, und sind am ehesten mit einer altertümlichen Pergamentrolle zu vergleichen, während ein Disketten- oder Plattenlaufwerk meist mit relativen Dateien arbeitet, was einem Karteikasten nahekommt.

Beide Verfahren haben ihre Vor- und Nachteile, die besonders im jeweiligen Platzbedarf und in der Zeit, die benötigt wird, um einen bestimmten Datensatz zu finden, zur Geltung kommen.
So wird jede Anschrift einer Kundendatei in einem Karteikasten eine eigene Karte beanspruchen, während auf der Rolle zwecks Papierersparnis alle Adressen möglichst dicht notiert wurden. Wird aber die Adresse des Kunden 234

gesucht, nimmt man sich die 234. Karte, denn selbstverständlich sind alle numeriert, bei der Rolle wird es sich hingegen nicht vermeiden lassen, alle Anschriften zu lesen oder sie zumindest von vorne an durchzuzählen.

Die Leser, die sich nun mehr mit Dateien befassen möchten, mögen dies mit entsprechender Literatur tun, ansonsten mag die Bemerkung genügen, daß die sequentielle Speicherung für unser Problem genau das richtige Verfahren ist, denn wir wollen unsere Variableninhalte der Reihe nach speichern und später wieder laden, das heißt, besondere Zugriffsmethoden sind für uns nicht erforderlich.

SAVE GAME

Die Übertragung der Daten wird zweckmäßigerweise innerhalb entsprechender Schleifen geschehen, deren Aufbau wir dem Initialisierungsteil unseres Adventures anlehnen. Dazu wird es jedoch notwendig sein, die Kenndaten des Programmes um die Anzahl der verwendeten Flags zu erweitern, denn auch dieser Schleife muß ein Endwert zugrunde gelegt werden.

Ebenso wird eine Änderung der Programmzeile 1500 erforderlich, um die neuen Routinen in die Abfrage der Sonderbefehle einzubauen:

```
160 AF=3

1500 IF LEFT$(EINGABE$,3)<>"INV" THEN GO
TO 1600
```

Handelte es sich bei der letzten Eingabe des Spielers nicht um den Befehl Inventur, werden einige Sprünge ausgeführt, bis die Identität mit einem der Kurzbefehle festgestellt worden ist:

```
1599 REM --------------------- SAVE GAME
```

```
1600 IF LEFT$(EINGABE$,4)<>"SAVE" THEN G
OTO 1700
1699 REM --------------------- LOAD GAME
1700 IF LEFT$(EINGABE$,4)<>"LOAD" THEN G
OTO 1900
```

Bevor das Betriebssystem unseres C64 dann die Daten übertragen kann, müssen ihm zunächst einige Informationen, wie Übertragungsrichtung, Transportweg und Adressat, genannt werden, eine Aufgabe, die der Openbefehl übernimmt.

```
1620 OPEN 2,8,2,"ATO:GAME,S,W"
```

Achten Sie bei der Eingabe bitte darauf, statt der Buchstaben AT die Taste rechts neben dem P zu drücken (Klammeraffe), ansonsten können Sie nur einen Syntax Error erwarten.

Nach Ausführung dieser Zeile steht auf der Diskette in dem Laufwerk mit der Gerätenummer 8 ein Datenfile ´Game´ zur Verfügung, welches sequentiell (S) beschrieben werden kann (W = Write). Das at - Zeichen wird erforderlich, damit ein fortgeschrittenes Spiel unter dem gleichen Namen abgespeichert werden kann.

Benutzer der Datasette geben statt dessen folgende Zeile ein,

```
1620 OPEN 1,1,1,"GAME"
```

und ändern jedes #2 in #1 um.

Der Übertragung von Spieldaten steht nach dieser Vorbereitung nichts mehr im Wege. Die Ausgabe der Daten an die Diskette erfolgt dabei in ähnlicher Form, wie eine

Ausgabe auf den Bildschirm, der Print Befehl muß nur durch die Angabe einer Filenummer umdirigiert werden.
Ein abschließender Close - Befehl übernimmt es, uns vor der Rückkehr ins Hauptprogramm vor später auftretenden Fehlern zu schützen.

```
1630 FOR I=1 TO AO
1631 PRINT #2,OB(I)
1632 NEXT I
1635 FOR RAUM=1 TO AR
1636 FOR RICHTUNG=1 TO 6
1637 PRINT #2, DURCHGANG(RA,RI)
1638 NEXT RICHTUNG
1639 NEXT RAUM
1645 FOR I=1 TO AF
1646 PRINT #2,FL(I)
1647 NEXT I
1650 CLOSE 2
1670 PRINT CHR$(147) : GOTO 1080
```

LOAD GAME

Zur Rekonstruktion einer bestimmten Situation ist es erforderlich, diese Daten wieder an die entsprechenden Variablen zu überweisen, eine Aufgabe, die ein fast identischer Programmteil übernimmt:

```
1720 OPEN 2,8,2,"GAME,S,R"
1725 INPUT #1,SPIELER
1730 FOR I=1 TO AO
1731 INPUT#1,OB(I)
1732 NEXT I
1735 FOR RAUM=1 TO AR
1736 FOR RICHTUNG=1 TO 6
1737 INPUT#2, DURCHGANG(RA,RI)
1738 NEXT RICHTUNG
```

```
1739 NEXT RAUM
1745 FOR I=1 TO AF
1746 INPUT #2,FL(I)
1747 NEXT I
1750 CLOSE 2
1770 PRINT CHR$(147) : GOTO 1080
```

Für die Verwendung der Datasette benutzen Sie folgende Zeile:

```
1720 OPEN 1,1,0,"GAME"
```

Natürlich müssen Sie ebenfalls wieder aus jedem #2 ein #1 machen.

Damit wäre die Aufgabe für Kassettensysteme gelöst, Diskettenbenutzer sollten allerdings ihre Möglichkeiten nutzen, und sich vor einer Fortsetzung des Spieles davon überzeugen, daß der Vorgang fehlerlos vonstatten ging.

Vielleicht befindet sich beim Aufrufen der Save - Funktion immer noch die Programmdiskette im Speicher, welche aus Sicherheitsgründen schreibgeschützt ist, oder es handelt sich um eine restlos beschriebene Diskette - viele Ursachen sind möglich, um die ganze vorangegangene Arbeit nutzlos zu machen.

Wie der Fehlerkanal ausgelesen wird, teilt uns das Bedienungshandbuch der Floppy auf Seite 36 mit, unsere Aufgabe ist es nun, die so erlangten Informationen auszuwerten und gegebenenfalls an den Spieler zu übermitteln.

Ergänzen wir unser Programm daher um folgende Zeilen,

```
1679 REM --------------- DISKETTENFEHLER
```

```
1680 OPEN1,8,15
1681 INPUT#1,A,B$,C,D
1682 IFA<>OTHENPRINT:PRINT"ACHTUNG:":PRI
NTB$:FORI=1TO5000:NEXT:CLOSE2:CLOSE1:GOT
O1080
1683 CLOSE1
1684 RETURN : REM ---- ENDE FEHLER -----
```

und rufen es an den notwendigen Stellen auf:

```
1660 GOSUB 1680
1760 GOSUB 1680
```

Um unser Programm noch luxuriöser auszustatten, werden wir noch einige weitere Zeilen ergänzen. So ist es durchaus nicht notwendig, daß wir uns auf ein einziges File zur Abspeicherung beschränken, eine zweite Datei wird sogar unbedingt erforderlich werden, wenn weitere Mitglieder unserer Familie ihr Herz für Adventures entdecken und sich an dem gleichen Spiel versuchen wollen.

Lassen wir den Spielern daher die frei Wahl und erlauben wir ihnen die Wahl unter allen denkbaren Namen bis zur maximalen Länge von sechzehn Buchstaben.

Ergänzen wir ferner noch einige Print- und Steuerbefehle, die dafür sorgen daß der saubere Aufbau des Bildschirmbildes erhalten bleibt.

Ein solchermaßen komplettiertes Unterprogramm wird dann ähnlich dem auf den folgenden Seiten abgedrucktem Listing aussehen.

```
1599 REM ---------------------- SAVE GAME
1600 IF LEFT$(EINGABE$,4)<>"SAVE" THEN G
OTO 1700
1605 PRINT"□"SPC(10)"SPIELSTAND SPEICHER
N":INPUT"▒▒UNTER WELCHEM NAMEN SPEICHERN
";EI$
1610 IF LEN(EI$)>16 THEN 1605
1615 PRINT"▒▒SPEICHERUNG VON ";EI$;" LAE
UFT ";
1620 OPEN 2,8,2,"§0:"+EI$+",S,W"
1625 PRINT#2,SPIELER
1627 PRINT#2, WERTUNG:PRINT#2,L1:PRINT#2
,LM:PRINT#2,LW
1628 PRINT#2, ZUG
1630 FOR I=1 TO AO
1631 PRINT#2,OB(I)
1632 NEXT I
1633 PRINT".";
1635 FOR RAUM=1 TO AR
1636 FOR RICHTUNG=1 TO 6
1637 PRINT#2, DURCHGANG(RA,RI)
1638 NEXT RICHTUNG
1639 NEXT RAUM
1640 PRINT".";
1645 FOR I=1 TO AF
1646 PRINT#2,FL(I)
1647 NEXT I
1648 PRINT".";
1650 CLOSE 2
1660 GOSUB 1680
1670 PRINT"□":GOTO 1080
1678 :
1679 REM --------------- DISKETTENFEHLER
1680 OPEN1,8,15
1681 INPUT#1,A,B$,C,D
1682 IFA<>OTHENPRINT:PRINT"▒▒▒▒ACHTUNG:
":PRINTB$:FORI=1TO 5000:NEXT:CLOSE2:CLOS
E1:GOTO1080
1683 CLOSE1
```

```
1684 RETURN : REM ---------- ENDE FEHLER
1698 :
1699 REM ------------------------ LOAD GAME
1700 IF LEFT$(EINGABE$,4)<>"LOAD" THEN G
OTO 1800
1705 PRINT"□"SPC(10)"ALTES SPIEL LADEN":
INPUT"▪▪WELCHEN SPIELSTAND LADEN";EI$
1710 IF LEN(EI$)>16 THEN 1805
1715 PRINT"▪▪"EI$" WIRD GELADEN.";
1720 OPEN 2,8,2,EI$+",S,R"
1725 INPUT#2,SPIELER
1726 PRINT".";
1727 INPUT#2, WERTUNG:INPUT#2,L1:INPUT#2
,LM:INPUT#2,LW
1728 INPUT#2, ZUG
1730 FOR I=1 TO AO
1731 INPUT#2,OB(I)
1732 NEXT I
1733 PRINT".";
1735 FOR RAUM=1 TO AR
1736 FOR RICHTUNG=1 TO 6
1737 INPUT#2, DURCHGANG(RA,RI)
1738 NEXT RICHTUNG
1739 NEXT RAUM
1740 PRINT".";
1745 FOR I=1 TO AF
1746 INPUT#2,FL(I)
1747 NEXT I
1748 PRINT".";
1750 CLOSE 2
1760 GOSUB 1680
1770 PRINT"□":GOTO 1080
1771 REM ---------------- ENDE LOAD GAME

READY.
```

DER WORTSCHATZ

Häufig macht die richtige Wortwahl ein Adventurespiel zur Qual, denn schließlich wird man als Spieler gezwungen, sich der gleichen Ausdrucksweise wie der Programmierer zu bedienen, was nicht immer ganz einfach ist.
Importe, die für den fortgeschritteneren amerikanischen Markt produziert wurden, lassen sich mit dem Schulenglisch häufig gar nicht, fast immer aber nur schwer lösen. Ein deutsch - englisches Wörterbuch hat bei einigen Spielern daher seinen festen Platz in Reichweite des Computers, doch auch dieses Hilfsmittel versagt, wenn sich der Autor der Umgangssprache oder eines Slangs bedient hat.

Scheitern viele Spiele an der Sprachbarriere, freut man sich um so mehr, wenn aus irgendeiner Quelle plötzlich ein deutsches Adventure auftaucht.
Doch sofort trüben zwei Eigenarten den rosaroten Blick, denn unsere Muttersprache ist bei weitem nicht so gut dazu geeignet, ähnlich wie bei den englischsprachigen Originalen, mit nur zwei Worten die unterschiedlichsten Situationen zu artikulieren. Schlimmes Gastarbeiterdeutsch mag noch akzeptabel sein, und auch über die unmöglichsten Satzkonstruktionen kann man zumindest zu Beginn noch lachen, doch wie soll ein normal begabter und veranlagter Mensch ohne Hilfestellung auf diese Eingaben kommen ?

´Nimm Messer´ ist dabei gar nicht einmal so übel, doch das Messer wird auch eine Aufgabe zu erfüllen haben. Es bleibt jedoch weiterhin ein erfreulicher Gegenstand, denn Eingaben wie ´Wirf Messer´, ´Schleife Messer´ oder ´Schärfe Messer´ erfordern nicht allzuviel Phantasie und sind ebenfalls recht eindeutig auszulegen.

Hat das Messer seine Funktion erfüllt, muß man sich seiner früher oder später entledigen, was bei einigen Spielen durch bewußtes Verlieren geschehen kann. Etwas besser als ´Verliere Messer´ wirkt schon ´Leg Messer´, doch taucht

automatisch die Frage nach dem wohin auf oder es wird zumindest noch ein ´weg´ erwartet. Diese Eingabe steht zwar im Widerspruch zu den üblichen Spielregeln, sollte aber dennoch, zumindest auf unserem System, versucht werden.

Betrachten wir abschließend noch einmal ein Beispiel der ersten Seiten dieses Buches. Wie gelangen wir in die Krone des Baumes, vielleicht mit ´Kletter Baum´ ?

Leider nicht, aber wohin bringt uns ein besser klingendes ´Erklettere Baum´ ?

Wieder nur die Mitteilung, daß das Wort ´Erklettere´ nicht verstanden wurde, also aufgeben oder weitergrübeln ?
Es wird sich nicht vermeiden lassen, Verben wie ersteige, besteige und erklimme zu erproben, und wenn die Aktion wirklich erforderlich ist, wird der Erfolg sich früher oder später einstellen.

Probleme dieser Art sind natürlich nicht sehr schön, und da dieses Kapitel von der Perfektionierung unseres Adventuresystemes handelt, werden wir uns nach geeigneten Lösungsmöglichkeiten umsehen.

SYNONYME

Gleichbedeutende Worte in reichlicher Menge einzuarbeiten, wäre ein erster Vorschlag zur Lösung des Sprachproblemes, wie jedoch das Beispiel des Baumes zeigt, kann die konsequente Ausstattung mit gleichbedeutenden Worten den Umfang eines Programmes sehr vergrößern, was ein immenses Maß zusätzlicher Routinearbeit bedeutet und den sowieso immer knapper werdenden freien Speicherplatz stark einschränkt.

Dennoch soll hier, wieder am Beispiel Goldrausch, ein Lösungsvorschlag eingebracht werden, der durch seine Kürze und Einfachheit doch schon fast wieder überzeugt. Selbstverständlich handelt es sich um keine zwingend erforderlichen Programmzeilen, es liegt an Ihnen, ob Sie Wert auf einen großen Wortschatz legen oder ob Sie sich mit dem anschließend gemachten Vorschlag zur Verbesserung der Benutzerfreundlichkeit zufrieden geben wollen.

Die erste Gelegenheit, nicht sofort das richtige Wort zu treffen, hat ein Spieler von Goldrausch in Raum 3 bei dem Versuch, die Holzhütte näher in Augenschein zu nehmen.
Bislang wird er mit ´Untersuche Holzhütte´ scheitern, der Erfolg wird sich erst bei ´Untersuche Hütte´ einstellen.

Damit der zweite Begriff überhaupt verstanden werden kann, müssen wir unser Vokabular zunächst erweitern:

```
120 AO=19
319 DATA "-","HOLZHUETTE",0
```

Die ausführliche Beschreibung können wir uns sparen, desgleichen wird die Null als Aufbewahrungsort verbindlich, schließlich soll weder im Inventory noch in einem Raum eine weitere Hütte auftauchen.
Benötigt wird nur der Rufname, damit der Adventuretreiber die entsprechende Objektnummer ermitteln kann, was in Zeile 2140 auch geschieht.

Normalerweise wird die Ausführung des Programmes in Zeile 2200 mit einem Sprung zur jeweiligen Befehlsausführung fortgesetzt. Dort findet sich natürlich keine einzige Bedingung, die erfüllt wäre, da nirgendwo für N ein Wert von 19 gefordert wird. Um diese Zeilen nicht ergänzen zu müssen, fügen wir folgenden Kunstgriff ein,

```
2140 IF EO$=RN$(N) THEN 2165
2164 REM -------------------- SYNONYME
```

```
2165 IF N=19 THEN N=4
```

und schon werden mit der Holzhuette genau die gleichen Aktionen durchgeführt wie mit der Huette.

Um unser Programm in die Lage zu versetzen, auch eine Vielzahl von Verben verstehen zu können, ist der zu treibende Aufwand sogar noch geringer.
Ein die gleiche Aktion einleitendes Wort wird ebenfalls dem Wortschatz des Adventures zur Verfügung gestellt, und die Sprungtabelle wird einfach um das bereits einmal benutzte Sprungziel ergänzt.

Goldrausch erweitern wir auf diese Weise um die zu ´Untersuche´ identische Funktion ´Betrachte´:

```
130 AV=7
207 DATA BETRACHTE
2200 ON VN GOTO 5000,6000,7000,8000,9000
10000,5000
```

Sie sehen, daß unser System sich um gleichbedeutende Verben besonders einfach erweitern läßt, weshalb wir zumindest auf diesen Bonus nicht verzichten sollten.

Synonyme der Objekte dürfen wir stiefmütterlich behandeln, denn ihnen kommt sowieso nicht die gleiche Bedeutung zu, denn die Kopfzeilen des Spieles werden fast immer, mit Ausnahme bei den Grafikadventures, Anhaltspunkte geben.

Wir können es dem Spieler aber auch einfacher machen, so daß er sich wirklich auf die Problemlösungen konzentrieren kann und auf der Suche nach den richtigen Worten nicht den roten Faden verliert.

Was würden Sie davon halten, wenn ein Adventure auf Wunsch sein gesamtes Vokabular preisgibt, wenn es eine Liste aller vorhandenen Objekte und möglichen Verben erstellt ?

```
ICH VERSTEHE FOLGENDE VERBEN

UNTERSUCHE
NIMM
LEG
OEFFNE
BENUTZE
ZERSTOERE
ZUENDE
FUELLE
BETRETE
LOESCHE
BEFESTIGE

                              TASTE DRUECKEN
```

VOKABULAR

Ein Bild wie das auf der vorangegangenen Seite dürfte alle Probleme der Wortwahl aus der Welt schaffen, und der Spieler verliert nicht durch sinnlose Eingaben Zeit und Lust.

Leider läßt sich an dieser Stelle zum zweiten Mal der Volksmund zitieren, denn 'Wo Licht ist, ist auch Schatten'.

Stellen wir uns einen Spieler vor, dem Goldrausch noch nicht bekannt ist. Ohne viel gespielt zu haben, wird er, wenn er einigermaßen konsequent und logisch denken kann, sogleich eine Lösungsstrategie entwickeln, die ihn schnell ans Ziel bringt. Er kennt sofort den gesamten Wortschatz und weiß, daß sich irgendwo eine Flasche finden läßt, daß irgendwo eine Truhe liegt und wird ebenfalls sogleich ein starkes Verlangen nach Silber und Nuggets entwickeln. Anschließend überlegt er, welche Gegenstände sich zerstören lassen, und seine Wahl wird auf die Flasche, die Hütte und die Kette fallen.

Er bemüht sich, die Schatztruhe zu finden, und nachdem er die Feststellung machte, daß eine Eisenkette ihn am Sichten des Inhaltes behindert, versucht er diese zu zerstören, was aber mit bloßen Händen kaum gelingen wird. Ein weiteres VOK wird ihm dann die Ahnung vermitteln, daß die Eisenstange vermutlich der einzige Gegenstand ist, der für diese Aufgabe in Frage kommt.

Aber auch die Gefahren werden entschärft, denn es bleibt ihm auch nicht verborgen, daß außer ihm noch ein Bär die Welt bevölkert !

Somit wird dem Programmierer auch noch die Aufgabe zufallen, genau abzuwägen, ob er den gesamten Wortschatz des Adventures preisgeben will oder nur eine Untermenge.

Dies könnte problemlos durch ein weiteres Titelbild geschehen, welches eine Anzahl der wichtigsten Worte ausgibt, allerdings würden wir damit unserem Ziel, ein universelles Adventuresystem zu entwickeln, entgegenhandeln, weil diese Tafel für jedes Programm neu erstellt werden müßte.

Für große Adventures mit einem entsprechenden Wortschatz bleibt die Auflistung des Vokabulars jedoch durchaus sinnvoll, weshalb wir bei der Entwicklung unseres Adventuresystemes ebenfalls nicht darauf verzichten wollen.

Die Behandlung des VOK - Befehles übernimmt aus eben genanntem Grunde der Treiber. Um eine wirklich universelle Verwendung dieses Programmteiles zu gewährleisten, müssen die zur Ausgabe bestimmten Worte aus den Spieldaten des betreffenden Adventures ermittelt werden.

Es wurde bereits kurz erwähnt, daß der Basic - Befehl Restore den sogenannten DATA - Zeiger auf den Beginn des

Datenblockes stellt, und das nächste Read wieder das erste Element der Liste einer Variablen zuweisen wird.

Glücklicherweise kommt die Anordnung unserer Adventuredaten einem solchen Vorgehen entgegen, da zuerst die Verben und daran anschließend die Objekte genannt werden, es muß keine Ausführungszeit auf das Lesen der übrigen Daten verwandt werden.

Bauen wir das entsprechende Unterprogramm, unter Berücksichtigung der wahrscheinlichen Häufigkeit eines Aufrufes der entsprechenden Zeilen, im Anschluß an die INV, SAVE, und LOAD Routine ein:

```
1800 IF LEFT$(EINGABE$,3)<>"VOK" THEN GO
TO 1900
1805 PRINT CHR$(147):PRINTCHR$(158);"ICH
VERSTEHE FOLGENDE VERBEN";CHR$(17);CHR$(
17)
1806 RESTORE
```

Ein nachfolgendes Read verhält sich nun genau so, wie der erste Read - Befehl nach dem Einschalten des Computers. Übernehmen wir daher die entsprechenden Programmzeilen aus Teil 1 mit der Änderung, daß die eingelesenen Daten nicht weiter gespeichert, sondern nur auf dem Bildschirm ausgedruckt werden:

```
1810 FOR I=1 TO AV
1820 READ VO$ : PRINT VO$
1830 NEXT I
```

Anschließend wollen wir dem Spieler ausreichend Zeit geben, die Verbliste zu studieren. Gleiches soll auch mit dem Ausdruck der Objekte geschehen, weshalb wir die entsprechenden Zeilen als Unterprogramm realisieren, welches an den betreffenden Stellen mit Gosub aufgerufen wird:

```
1840 GOSUB 1890
1845 PRINTCHR$(147):PRINT"UND FOLGENDE O
BJEKTE SIND MIR BEKANNT:"CHR$(17)
1850 FOR I=1 TO AO
1860 READ VO$,VO$,X : PRINT VO$
1870 NEXT I
1880 GOSUB 1890:PRINT CHR$(147):GOTO 108
0

1890 PRINTSPC(24);CHR$(17);"TASTE DRUEC
KEN";
1895 GET EINGABE$ : IF EINGABE$="" THEN
1895
1896 PRINT CHR$(147) : RETURN
```

Zu beachten ist, daß ohne die Verwendung entsprechender Basic - Erweiterungen, DATA - Zeilen ebenfalls sequentiell gelesen werden. Zeile 1860 weist der Stringvariablen VO$ daher zunächst die ausführliche Objektbeschreibung zu, die, weil wir diesen langen Text an dieser Stelle nicht benötigen, daran anschließend sofort mit dem Rufnamen überschrieben wird.

Zum Abschluß der Auflistung des gesamten Vokabulars wird wieder das Unterprogramm ab Zeile 1890 aufgerufen, damit nach Drücken einer weiteren Taste das Adventure fortgesetzt werden kann.

Obige Zeilen werden nun wie vorgesehen arbeiten, zumindest solange nicht mehr als zwanzig Worte ausgegeben werden müssen. Nach überschreiten dieser Zahl wird die Verweildauer der ersten Vokabeln auf dem Bildschirm kaum von ausreichender Länge sein, um eine Hilfe für den Spieler zu sein.

Zur Lösung des Problemes führen wir die Kontrollvariable Zeile ein. Sie wird zu Beginn der Ausgabe auf Null gesetzt, und bei der Ausgabe eines jeden Wortes um eins erhöht. Sind mit zwanzig Worten entsprechend viele Zeilen benutzt worden, löschen wir den Bildschirm und setzen den Zähler wieder auf eins zurück.

```
1849 ZEILE=0
1850 FOR I=1 TO AO
1855 ZEILE=ZEILE+1
1860 READ VO$,VO$,X : PRINT VO$
1865 IF ZEILE=20 THEN GOSUB 1890
1866 IF ZEILE=20 THEN ZEILE=1 : PRINT CH
R$(147)
1870 NEXT I
```

Ein Programmlauf bringt nun das gewünschte Ergebnis, doch entsprechen die erscheinenden Kürzel UNT, OEF und wie sie alle heißen keineswegs dem Standard unserer Programme.

Um ein zufriedenstellenderes Ergebnis zu erzielen, ändern wir die Verben und Objekte in den für sie vorgesehenen DATA - Zeilen (200 - 500) und machen uns die Mühe, sie, ohne Berücksichtigung der relevanten Wortlänge, auszuschreiben.

Durch diese Manipulation ergibt sich nun die Notwendigkeit einer weiteren Ergänzung in den Zeilen 815 und 835, schließlich wollen wir den für die Variablen vorgesehenen Speicherplatz nicht unnötig verschwenden.

Warum soll das ganze Wort gespeichert werden, wenn zur eindeutigen Identifizierung eines Spielzuges nur drei oder vier Buchstaben von Bedeutung sind.

```
815 READ VERB$(I):VERB$(I)=LEFT$(VERB$(I
),WL)
835 READ OB$(OBJEKT), RN$(OBJEKT), OB(OB
JEKT):RN$(OB)=LEFT$(RN$(OB),WL)
```

HELP

Die Entwicklung einer Help - Routine soll nun ein letzter Schritt sein, um die Benutzerfreundlichkeit unserer Adventures zu erhöhen.

Glaubt der Spieler, daß er sich restlos verfahren hat, so wird er versuchen, durch die bloße Eingabe von Help in den Besitz hilfreicher Informationen zu gelangen.

Handelt es sich um ein einfaches Spiel, weil im Verlauf desselben genügend Hinweise gegeben werden, machen wir uns die Sache einfach:

```
1959 REM ------------------------- HELP
1960 IF LEFT$(EINGABE$,4)<>"HELP" THEN G
OTO 2000
1970 PRINT"ICH KANN DIE SPIELREGELN WIED
ERHOLEN.":PRINT"WUENSCHEN SIE DAS ?"
1971 GET EI$:IF EI$="" THEN 1971
1972 IF EI$="J" THEN GOSUB 900
1975 GOTO 1080
1979 REM -------------------- ENDE HELP
```

Eine noch häufiger anzutreffende Standardantwort lautet ´Ich würde alles untersuchen ! - Try examining things !". Meist werden jedoch, zumindest in den Abenteuerspielen neueren Datums, Ratschläge gegeben, die manchmal allerdings wenig hilfreich, weil für den Spieler nicht zu verstehen, sind.

Denn da es sich um einen Ein - Wort Befehl handelt, steht die gewünschte Hilfe nicht in Beziehung zu einem bestimmten Objekt, was eine Auswertung der Situation in Hinblick auf die Schwierigkeiten des Spielers nicht gerade einfach macht.

Schließlich steht dem Programm als Arbeitsgrundlage nur der Aufenthaltsraum des Spielers zur Verfügung, und so ist es nicht weiter verwunderlich, daß viele Adventures in problematischen Räumen immer wieder die gleiche Mitteilung an den Abenteurer ausgeben, die er deshalb nicht verstehen kann, weil seine Gedanken bislang um ein anderes Problem kreisen.

Als typisches Beispiel ziehen wir eine Spielszene aus Raum 4 heran. Der Spieler hat die Schatztruhe entdeckt und müht sich seit längerer Zeit vergeblich mit der hinderlichen Eisenkette ab. In dieser Situation wäre folgender Hinweis denkbar:

```
1970 IF SP=4 THEN PRINT"EIN VERLAENGERTE
R ARM VERSTAERKT DIE KRAFT.":GOTO1080
```

Was könnte ein Spieler, der in Raum 4 um Hilfe bittet, mit der Information, an die Hebelgesetze zu denken, anfangen, wenn er bereits bei der Suche nach der Truhe gescheitert ist und er somit die Kette noch nicht entdeckt haben kann ?

Will man den Spieler verwöhnen und in besonders schwierigen Situationen Schritt für Schritt an die Lösung heranführen, wird es, wie obiges Beispiel zeigt, nicht ohne die Formulierung umfangreicher Tests und Bedingungen gehen. Für diese Aufgabe scheinen wieder einmal die Flags wie auch die Aufbewahrungsorte der verschiedensten Gegenstände prädestiniert zu sein.

Wiederum am Beispiel der Truhe probieren wir einmal diese Art der genau dosierten Hilfestellung aus, was allerdings nicht heißen soll, daß diese Szene als besonders schwierig zu gelten hat:

```
606 MS$(6)="WIE KANN ICH DIE KETTE ZERST
OEREN ?"
```

```
1970 IF SP=4 AND OB(10)=0 THEN PRINT"FAS
T WAERE ICH IN EINE GRUBE GEFALLEN.":GOT
O 1080
1971 IF SP=4 AND OB(11)=SP AND NOT FL(2)
THEN PRINTMS$(6);:PRINTMS$(7):GOTO1080
```

Erinnern wir uns, daß beim Passieren des betreffenden Raumes zunächst nur die Büsche sichtbar sind, und das Erdloch zwischen den Büschen erst entdeckt werden muß. Erlangt der Spieler jedoch die Information eines Beinahe - Sturzes in eine Grube, wird er sich wohl näher mit diesem Hindernis beschäftigen.

Ein weiterer Versuch mit Help bringt zu diesem Zeitpunkt keinen besonderen Vorteil. Erst nachdem die Truhe als sichtbares Objekt zur Ausstattung des Raumes gehört, erhält der Spieler den Hinweis, daß die Begutachtung des Inhaltes die Zerstörung der Kette und das Öffnen des Deckels erfordert.

Diese Aktionen wird er dann wohl hoffentlich alleine durchführen können, falls nicht, wäre es für diesen Spieler eine Überlegung wert, ob er nicht lieber einen Roman lesen oder mit Arcade - Action Spielen befassen sollte.

Den Abschluß einer solchermaßen durchkonstruierten Hilfestellung wird wieder eine Sicherheitszeile, wie wir sie bereits bei der Programmierung der anderen Aktionen kennengelernt haben, bilden, eine Zeile, die immer dann ausgeführt wird, wenn keine speziell definierte Situation eingetreten ist:

```
1971:
1972:
1975 PRINT"ERST SEHEN, DANN DENKEN UND Z
ULETZT HANDELN !": GOTO 1080
```

MOTIVIERUNG DES SPIELERS

Unser nächster Vorschlag für bessere Adventures wird unseren Adventuretreiber um einen letzten Ein - Wort Befehl erweitern.

Abenteuerprogramme benötigen für ihre Lösung nun einmal einen gewissen Zeitaufwand, wenn man jedoch tagelang spielt und überhaupt keinen Fortschritt sieht, fragt man sich eines Tages doch, ob man nicht ein anderes Spiel laden soll.

Aus diesem Grunde halte ich Adventures, die mit einer Punktwertung arbeiten und dem Spieler anzeigen, wie weit er noch von seinem endgültigen Ziel entfernt ist, für bedeutend reizvoller.

Jedoch gilt es zu überlegen, auf welcher Basis die Wertung durchgeführt werden soll. Die übliche Version sieht für jeden gefundenen Schatz beziehungsweise für jede gelöste Teilaufgabe in einem Mission - Adventure eine gewisse Anzahl von Punkten vor. Neben der Angabe ´Von 100 möglichen Punkten hast du 40´ wird dem Spieler auch noch der prozentuale Anteil mitgeteilt, wodurch der Spielstand einfacher und genauer ausgewertet wird. Als Beispiel kann wieder einmal Scott Adams ´Adventureland´ angeführt werden, dreizehn Schätze ergeben eine krumme Anzahl von Punkten, bei mehr als 50 Prozentpunkten weiß der Spieler jedoch, daß er die Hälfte der Strecke geschafft hat.

Eine alternative Lösung wird dagegen jeden Schritt voran belohnen und auch nicht davor zurückschrecken, für jede geleistete Hilfe eine gewisse Anzahl von Punkten abzuziehen. Ebenso wird jeder Fehler mit Todesfolge negativ zu Buche schlagen, so daß im extremen Fall sogar ein negativer Wert erzielt werden kann. Doch ist es gerade dieser große Spielraum, der zu wiederholten

Lösungsversuchen reizt, denn es müßte doch möglich sein, mehr Punkte zu erzielen als der Freund, der das Ziel ebenfalls erreicht hat.

Soll ein Adventure dermaßen ausgestattet werden, beginnt die Arbeit des Programmierers mit der Auswahl geeigneter Sequenzen, schließlich soll nicht bereits das einfache Auffinden eines Gegenstandes Punkte wert sein. Eine Belohnung darf der Spieler jedoch erwarten, wenn er gefährliche Monster ausgetrickst oder geheime Durchgänge gefunden hat.

Bevor wir uns nun detailliert mit der praktischen Ausführung entsprechender Versionen befassen, vermitteln wir als erstes dem Treiber das Verständnis für den Befehl ´SCORE´:

```
1500 IF LEFT$(EINGABE$,3)<>"INV" THEN GO
TO 1560
1559 REM -------------------------- SCORE
1560 IF LEFT$(EINGABE$,3)<>"SCO" THEN GO
TO 1600
```

Anschließend führen wir zur Speicherung des erzielten Punktewertes die Variable WERTUNG ein. Dabei ist es wichtig, dafür Sorge zu tragen, daß sie bei jedem Neustart, somit auch beim Tode der Hauptfigur, auf Null gesetzt wird.

Nur der erreichte Stand des Punktekontos sagt natürlich gar nichts aus, wenn er nicht in Beziehung zur maximalen Punktzahl steht, deshalb:

```
170 WMAX=20 :REM MAXIMALE PKTE DER MINI
VERSION
```

```
171 WERTUNG=0 :REM 0 PUNKTE
1561 PRINT"VON";WMAX;"PUNKTEN HAST DU"
;WERTUNG;"PUNKTE."
1565 GOTO 1080
```

Die Punkte erzielt der Spieler mit dem Aufnehmen der Gold- und Silberstücke. Ergänzen wir darum die betreffenden Zeilen in unserem Aktionsteil:

```
6014 IF N=12 THEN PRINT"O.K.":OB(12)=-1:
WE=WE+10:GOTO1080
6017 IF N=17 AND FL(3) THEN PRINT"O.K.":
OB(17)=-1:WE=WE+10:GOTO 1080
```

Leider kann ein Spieler, der es nur auf Punkte abgesehen hat, mit der derzeitigen Ausführung des Programmes beliebig hohe Summen erlangen.

Erinnern wir uns, daß wir bei der Ausgestaltung der Aktion ´Nimm´ darin übereingekommen waren, daß mit nehmen ein ´in die Hand nehmen´ gemeint war. Dies ermöglichte der Spielpraxis neben der Aufnahme von Gegenständen zwecks Aufbewahrung im Inventory auch ein Nehmen aus dem Inventory, um eine Aktion, beispielsweise das Öffnen einer Tür mittels eines Schlüssels, durchzuführen.

Gerade diese Auslegung erweist sich nun als hinderlich, nichts hält den Spieler davon ab, mehrmals ´Nimm Silbermuenzen´ einzugeben und mit dieser Eingabe jeweils zehn Punkte auf seinem Konto hinzuzuaddieren.

Als Ausweg bieten sich einmal eine Änderung der allgemeinen Vorbedingung (Zeile 6000) wie auch die Einführung zusätzlicher Bedingungen in die Aktion selbst, an.
Dieses Prinzip kennen wir bereits von der Ausführung der Aktion ´Untersuche´, hier mußte eine ähnliche Regelung

verhindern, daß ein Gegenstand immer wieder an seinem Ursprungsort sichtbar wurde.

```
6014 IF N=12 AND OB(N)=4 THEN PRINT"O.K.
":OB(12)=-1:WE=WE+10:GOTO1080
6017 IF N=17 AND OB(N)=5 AND FL(3) THEN
PRINT"O.K.":OB(17)=-1:WE=WE+10:GOTO 1080

6020 IF N=12 AND OB(N)=-1 THEN PRINT"DAS
SILBER HABE ICH BEREITS !":GOTO1080
6021 IF N=17 AND OB(N)=-1 THEN PRINT"ICH
BIN BEREITS IM BESITZ DES GOLDES !":GOTO
1080
```

Entscheiden Sie sich hingegen, die Aktion ´Nehmen´ immer nur als Bereicherung des Spielers aufzufassen, können Sie sich auf eine Verschärfung der Eingangsbedingung und Ausgabe eines entsprechenden Hinweises beschränken:

```
6000 IF N<>SP THEN GOTO 6900
6998 PRINT"DAS HABE ICH DOCH SCHON !":GO
TO 1080
```

Für unsere eigenen, das heißt zumindest für die Adventures dieses Buches, wollen wir nur eine Punktierung der in den Besitz des Spielers gebrachten Schätze vornehmen. Dennoch werden wir nicht auf eine Bewertung der einzelnen Züge des Spielers verzichten müssen.

Im Gegenteil, wir werden sogar noch einen Schritt weiter gehen und jeden einzelnen Spieler bewerten, ihm die erreichte Durchschnittswertung mitteilen.

Als Grundlage für diese Auswertung bietet sich die Anzahl der benötigten Züge an. Ein erfahrener Abenteurer wird sich

nicht mit der Untersuchung jeder Kleinigkeit aufhalten, ebenso weiß er, aus typischen Hinweisen seinen Nutzen zu ziehen.
Aus dieser Summe und den dabei erreichten Punkten berechnen wir einen Durchschnittswert und nehmen anhand dieses Quotienten die Beurteilung des Spielers vor.

```
180 ZUG=0 : WERTUNG=0

1080 PRINT CHR$(154) : ZUG=ZUG+1

1561 PRINT"VON";WMAX;"PUNKTEN HAST DU
IN";ZUG;"ZUEGEN"
1562 WERTUNG;"PUNKTE ERREICHT ! DAS ENT
SPRICHT EIN"
1563 PRINT"SCHNITT VON";WERT/ZUG;"PUNK
TEN."
1565 GOTO 1080
```

Auch das Kriterium für ein erfolgreiches Spielende definieren wir neu. Anstelle einer genau definierten Bedingung, die spielabhängig ist, vergleichen wir nun die erreichte Punktzahl mit dem maximal möglichen Wert. Ist die Differenz gleich Null, hat der Spieler die Aufgabe gelöst.

Zweckmäßigerweise erfolgt dieser Test gleich zu Beginn eines jeden Zuges, bevor überhaupt irgendwelche Daten an den Spieler übermittelt werden.

```
1085 IF WERTUNG=WMAX THEN GOTO 4800
```

Falls Sie ebenfalls die Punktzahl als Kriterium verwenden, vergessen Sie bitte nicht, alle anderen Zeilen, die mit der gleichen Aufgabe beschäftigt sind, aus dem Programm zu entfernen (Zeilennummern im Bereich von 1331 bis 1389).

Mit der konsequenten Einführung eines Punktesystems stellt sich uns zum Abschluß noch die Aufgabe, den Siegestitel an diesen Aufbau anzupassen. Die ermittelten Daten sollten noch einmal zusammengefaßt werden, damit auch eine Benotung des Spielers stattfinden kann.

Dazu wird es sich allerdings nicht vermeiden lassen, daß Sie, wenn das Adventure fertiggestellt ist und keinerlei Fehler mehr aufweist, einen kompletten Lauf durchspielen. Bei diesem Spiel verzichten Sie auf alle nicht unbedingt erforderlichen Eingaben und ermitteln die Anzahl der für einen Sieg mindestens erforderlichen Züge. Berechnen Sie das Verhältnis zwischen Punkte und notwendigen Eingaben und legen Sie diesen oder einen etwas geringeren Wert als Kennzeichen für ein sehr gutes Spiel zugrunde.
Wählen Sie entsprechend abgestufte Zahlen für schlechtere Beurteilungen.

So kann die Benotung eines Spielverlaufes der Miniversion von Goldrausch etwa folgendermaßen vorgenommen werden:

```
4800 PRINT CHR$(147)
4805 EW=WERTUNG/ZUG
4810 PRINT"HERZLICHEN GLUECKWUNSCH !"
4815 PRINT"SIE HABEN DIE IHNEN GESTELLTE
AUFGABE"
4820 PRINT"GELOEST."
4825 PRINT"IN ";ZUG;" SPIELZUEGEN HABEN
SIE PRO ZUG"
4830 PRINT EW;" PUNKTE GEMACHT."
4835 PRINT"DAMIT HABEN SIE EIN ";
4840 IF EW<0.5 THEN MS$="MISERABLES"
4845 IF EW>0.5 THEN MS$="MAESSIGES"
4850 IF EW>1.5 THEN MS$="SEHR GUTES"
4855 IF EW>1.0 THEN MS$="GUTES"
4860 PRINT MS$(0);" ERGEBNIS"
4865 PRINT"ERZIELT."
4899 END
```

Ein solchermaßen gestalteter Titel, entsprechend aufgemacht durch die Verwendung von Steuer-, Farb- und Grafikzeichen, wird manchen Abenteurer, der das Adventure endlich gemeistert hat, dazu veranlassen, es noch einmal zu spielen.

Und daß alles nur, um als guter Spieler bezeichnet zu werden.

Mit diesen Bemerkungen wäre das Thema ´Score´ eigentlich beendet, wenn wir nicht noch eine technische Ergänzung machen müßten, deren Fehlen ansonsten die ganze schöne Planung zunichte machen kann und mit Sicherheit die Spieler unserer Programme verärgern würde.

Vielleicht haben Sie bereits an unsere Save Game Routine gedacht, der natürlich auch ein Abspeichern und Laden der oben eingeführten Daten Wertung und Zug ermöglicht werden muß, wenn der Spieler nicht immer wieder mit null Punkten beginnen soll.

```
1727 PRINT#2, WERTUNG
1728 PRINT#2, ZUG

1827 INPUT#2, WERTUNG
1828 INPUT#2, ZUG
```

Alle bisher in diesem Kapitel gemachten Vorschläge dienten einzig und allein dem Ziel, den Komfort der Programme zu erhöhen.

Ich glaube, daß Sie mit mir einer Meinung sind, wenn ich die Behauptung aufstelle, daß Adventures, die nach dem Konzept dieses Buches entwickelt wurden und sämtliche

Extras enthalten, durchaus zur Oberklasse der Textadventures gezählt werden dürfen. Schließen wir daher die technische Entwicklung mit diesen Zeilen ab, und wenden wir uns der Frage zu, wie wir die Spiele selbst attraktiver und auch schwieriger machen können.

Dabei werden wir es halten wie gewohnt, nach trockener Theorie folgen entsprechende Programmzeilen, die wiederum für unser Erstlingswerk Goldrausch gedacht sein werden.

Die folgenden Seiten werden Ihnen insbesondere zeigen, wie Sie dem Spieler weitere Steine in den Weg legen können, so daß er tatsächlich die angebotenen Kurzbefehle Save und Load nutzen muß, um irgendwann einmal ans Ziel zu gelangen.

Um ausreichend Platz für Monster, Falltüren und was es sonst noch gibt, zu haben, werden wir uns als nächstes wieder einmal mit der Geografie befassen und unsere Miniwelt zur Grundlage eines ausgewachsenen Adventures entwickeln.

Falls Sie allerdings beabsichtigen, selber in den Genuß eines Adventurespieles zu kommen, sollten Sie, bevor Sie sich mit den nächsten Abschnitten befassen, alle noch fehlenden Programmzeilen eingeben oder besser noch, von jemand anderem eingeben lassen.

Ein entsprechendes Listing, welches alle gemachten Vorschläge berücksichtigt, finden Sie im nächsten Kapitel.

ORIENTIERUNG ODER DESORIENTIERUNG

Es wurde bereits zu Beginn des Buches erwähnt, daß die meisten Abenteuerspiele sich innerhalb einer Welt von ungefähr vierzig Räumen abspielen. Dennoch entsteht der Eindruck einer viel größeren Welt, gab es da doch den

riesigen Wald, einen Sumpf, eine unterirdische Höhlenlandschaft, ein Labyrinth und und und .

Tatsächlich sind jedoch gerade die so weiträumig erscheinenden Gebiete Gruppen von einigen wenigen Räumen, die auf so geschickte Art und Weise miteinander verbunden wurden, daß der Spieler meistens nicht einmal bemerkt, auf welche Tour er da geschickt wurde.

Die Palette der Möglichkeiten reicht dabei vom einfachen Raum, der immer wieder in sich selbst mündet, bis zu einer Kette von Räumen, an deren Ende der Spieler wieder an den Beginn versetzt wird.

Nutzen wir an dieser Stelle die in Goldrausch ähnlichen Räume 1 und 2, um gleich zu Spielbeginn ein weitläufiges Areal zur Verfügung zu haben.
Lassen Sie Raum 1 in nördlicher und westlicher Richtung wieder in sich selbst zurückmünden, verfahren Sie gleichermaßen mit Raum 2, und versuchen Sie dann, sich den Spieler vorzustellen, der in Raum 1 beginnt und nach Westen wandert. Woher soll er wissen, daß er sich nach jeder Eingabe immer noch im gleichen Teil des Waldes befindet ?
Falls er nicht das Glück hat, das Spiel mit der Eingabe ´O´ zu beginnen, wird er eine Reihe von Eingaben benötigen, um den Weg aus dem Wald zu finden, was ihm einen guten Schnitt bei der Punktwertung bereits verdirbt.

```
501 ... 1,1,1,2,0,0
502 ... 2,2,3,1,0,0
```

Noch schwieriger wird sich die Suche nach dem rechten Weg gestalten, wenn der südliche Ausgang von Raum 2 den Spieler nicht wieder in Raum 2 gehen, sondern ihn Raum 1 betreten läßt.

```
501 ... 1,1,1,2,0,0
```

502 ... 2,1,1,3,0,0

Starten Sie das Programm, und lassen Sie sich von der Wirkung dieser kleinen Änderung überraschen.

Mit drei Räumen lassen sich auf diese Art und Weise bereits kleine Labyrinthe erstellen, wenn die Verbindungen wahllos, ohne Bezug zu einem Kompaß, ausgeführt werden, und nur ein einziger Zu- wie auch Abgang existieren.

Die übliche, bislang auch von uns verwandte Verbindungstechnik hält sich an die realen, durch Himmelsrichtungen vorgegebenen Wege. Dabei entsteht eine Welt, die eine problemlose Identifizierung der Räume und, besonders wenn der Spieler eine entsprechende Karte anfertigt, Orientierung gestattet.

Kleine Sprünge über einige Räume hinweg machen dieses Konzept jedoch schnell unbrauchbar.
Stellen wir uns drei hintereinander liegende Räume vor, die alle mit der Beschreibung ´Ich bin in einer großen Höhle´ aufwarten.
Ein Spieler, der von Westen in Höhle 1 gelangt, wird, da er nichts besonderes sieht, zunächst mit großen Schritten die nähere Umgebung kennenlernen wollen und vermutlich seine einmal eingeschlagene Richtung nicht ändern, da es ihm ein Bedürfnis ist, möglichst einfach den Weg zurückzufinden.

Nach Durchqueren von Raum 2 und 3 gelangt er wiederum in den ersten Raum, was er ohne entsprechende Handlungen aber unmöglich feststellen kann.

Durch Verwendung dieser Spungtechnik ist es möglich, ohne großen Aufwand riesig erscheinende Labyrinthe zu erstellen.
Sechs Räume sind völlig ausreichend, dem Spieler das Finden des richtigen Weges fast unmöglich zu machen.
Selbst der Programmierer wird während der Testphase auf ausreichende Schwierigkeiten treffen !

Denn selbst die Anfertigung einer Karte wird bei dieser Arbeitsweise zum Problem, nicht jedoch bei einem alternativem Verfahren, das an dieser Stelle auch kurz vorgestellt werden soll.

Eine Vorgehensweise nach dieser Methode sieht zunächst ein maßstabsgerecht auf vorzugsweise kariertem Papier erstelltes Labyrinth vor, wie Sie es häufig auch in Rätselecken der verschiendensten Zeitschriften finden.
Bei der anschließenden Programmierung entspricht dabei jedes Kästchen einem Raum, wodurch der beträchtliche Aufwand bereits abzusehen ist.
Ebenso leicht ist es einzusehen, daß der Spieler während seiner Exkursion mit jedem neuen Raum ein neues Kästchen nimmt, die Wände markiert und die Durchgänge blank läßt.

Nehmen Sie stattdessen sechs Räume, von denen Sie jeweils einen als Eingangs-, als Sammel- und als Ausgangsraum festlegen. Dabei ist es für den Eingangsraum wichtig, daß er allein und auch nur in einer einzigen Richtung wieder zurück an den zuletzt besuchten Ort führt.
Entsprechendes gilt für den letzten Raum, nur ein einziger Durchgang erlaubt das Betreten von Neuland, alle anderen Wege müssen ins Labyrinth zurückführen.
Dem Sammelraum kommt dabei eine zentrale Funktion zu. Von jedem Raum des Labyrinthes führen zwei oder mehr Wege in diesen Raum, alle Ausgänge bringen den Spieler jedoch wieder in den ersten oder auch zweiten Raum des Irrgartens.

Um die Sache noch zu komplizieren, werden die Ausgänge des Raumes, der verbindungsmäßig vor dem letzten Raum liegt, noch in der Art angelegt, daß nur ein einziger Weg in den letzten, alle anderen aber in den Sammel- oder in den ersten Raum führen.
Es erübrigt sich zu sagen, daß für alle Räume eine identische Beschreibung vorgesehen wird und daß möglichst alle sechs Richtungen benutzt werden.

Der Spieler wird nun fast soviel Glück wie ein Lottosieger benötigen, um das Labyrinth verlassen zu können: ehe er im vorletzten Raum aus sechs Wegen den einzig richtigen findet, gleiches gilt für den letzten Raum. Dabei wirft ihn jeder Fehler wieder in einen anderen Raum zurück, welcher wiederum sechs Möglichkeiten bietet.

Versuchen Sie, sich im folgenden Labyrinth zurechtzufinden - obwohl Sie selbst die Verbindungen erst kurze Zeit zuvor eingegeben haben, wird es Ihnen schwer fallen:

Das Labyrinth von Goldrausch:

```
110 AR=14

507 DATA "AM EINGANGSSTOLLEN EINES ALTEN
 BERGWERKES.",7,0,1,5,0,0
519 DATA AUF EINEM KURVENREICHEN GANG.,8
,0,0,5,0,0
524 DATA AUF EINEM KURVENREICHEN GANG.,1
0,7,8,8,8,8
525 DATA AUF EINEM KURVENREICHEN GANG.,1
1,8,8,13,11,8
526 DATA AUF EINEM KURVENREICHEN GANG.,1
1,8,10,11,11,8
527 DATA AUF EINEM KURVENREICHEN GANG.,9
,8,10,11,10,8
528 DATA AUF EINEM KURVENREICHEN GANG.,8
,10,10,11,0,0
529 DATA AUF EINEM KURVENREICHEN GANG.,0
11,12,11,0,0
```

ORTSWECHSEL

Außer Irrgärten in beliebiger Größe stehen uns weitere Mittel zur Verfügung, um dem Spieler die Orientierung zu erschweren.

Sehr großer Beliebtheit erfreuen sich dabei plötzliche Transfers der Hauptperson in einen anderen Raum, was dem Abenteurer durch Mitteilungen wie ´Alles um mich herum dreht sich´ deutlich gemacht wird.
Doch ist auch diese Nachricht bestenfalls als freundliche Geste des Programmierers aufzufassen, und nicht als Pflicht zu verstehen.

Auslöser der Aktionen solcher Art sind meist magische Gegenstände, zu deren Bedienung seit Aladin´s Wunderlampe ein kräftiges Reiben erforderlich ist.
Aber auch Zaubersprüche und -tränke verfehlen ihre Wirkung nicht, weshalb dem Spieler eines Adventureprogrammes nur geraten werden kann, allergrößte Vorsicht walten zu lassen, wenn in irgendeinem Zusammenhang das Wort ´Magie´ fallen sollte.

Programmtechnisch werfen diese Ortswechsel bei Verwendung unseres Adventure - Systems keinerlei Schwierigkeiten auf, denn es reicht aus, der Variablen SP die Nummer des neuen Raumes zuzuweisen.

´Reibe Lampe´ könnte daher folgendermaßen implementiert werden (Lampe = Objekt 3):

```
xxxx IF N=3 AND OB(3)=-1 THEN PRINT"DIE
WELT DREHT SICH ...":SP=9:GOTO 1080
```

Wenn dann auch noch die Raumbeschreibung von 9 der des alten Raumes entspricht, dürfte die Verwirrung des Spielers nach dem nächsten Zug unausweichlich bleiben. Dies gilt

besonders für den Fall, wenn die Meldung fortfällt oder ein anderer Text gewählt wird:

```
xxxx IF N=3 AND OB(3)=-1 THEN PRINT"O.K.
 - NICHTS PASSIERT.":SP=9:GOTO 1080
```

Eine weitere Steigerung des Schwierigkeitsgrades bedeutet es, wenn der Ortswechsel des Spielers nicht einmal mehr von den Manipulationen irgendwelcher Gegenstände abhängig ist, sondern einzig und allein vom Zufall gesteuert wird.

Wir können mit der Konstruktion unseres Adventureprogrammes vorschreiben, daß das Durchqueren eines bestimmten Raumes unumgänglich ist.
Entspricht der Inhalt von SP der Nummer dieses Raumes, wird eine zufällige Entscheidung darüber getroffen, ob der Spieler versetzt wird oder ob ihm zwecks Ausführung weiterer Handlungen ein Aufenthalt gestattet wird.

Welcher Spieler, der nach erfolgter Durchquerung eines Irrgartens und anschließendem Betreten eines scheinbar harmlosen Raumes mehr als einmal wieder in das Labyrinth zurückgeworfen wurde, wird das Risiko einer Wiederholung dieses Vorganges eingehen ?

Wieder zeigt ein Ausschnitt aus dem Adventure Goldrausch, wie dieses Handicap eingebaut werden kann:

```
110 AR=16

519 DATA AUF EINEM KURVENREICHEN GANG.,8
,0,15,5,0,0
521 DATA IN EINEM STOLLEN.,11,11,11,7,11
,0
522 DATA IN EINEM FELSENDOM.,0,15,0,0,0,
0
1340 IF SP=15 THEN IF RND(1)>.7 THEN SP=
16
```

Versetzt in den kritischen Raum wird der Spieler durch Verlassen des Vorraumes zum Labyrinth, zur Zeit Raum acht, ein kurvenreicher Gang, in westlicher Richtung.

Aus diesem Raum kann er sich sofort wieder zurückziehen, oder er kann versuchen, herauszufinden, was die weitere Umgebung bringt.

Wenn der Zufall es will - und ob er will, legen wir mit dem Vergleichswert in der RND - Funktion fest -, gelangt er in den nächsten Raum (16), wenn nicht, findet er sich in einem Raum des Labyrinthes (Sammelraum 11) wieder, welcher ihm keine Anhaltspunkte zur Orientierung bietet.

War ihm das Glück nicht hold, wird er nach einigen Versuchen die Entscheidung treffen, daß es sich um einen weiteren Zugang des Irrgartens handelt, diesen Raum fortan meiden und somit einen Teil des Schatzes niemals finden.

VERSTECKTE ZUGÄNGE

können dem Spieler mindestens ebenso große Schwierigkeiten bereiten. Selbst wenn in der Zeile ´Ich kann nach ...´ nur die Richtungen Osten und Westen genannt werden, muß deshalb die Welt im Norden nicht zu Ende sein:

```
ICH SEHE EINE DUESTERE FELSENHOEHLE,
EINEN GRIMMIG DREINBLICKENDEN BAEREN,
*NUGGETS*.
ICH KANN NACH WESTEN, OSTEN.
________________________________________

O.K.
WAS SOLL ICH TUN? O
O.K.
WAS SOLL ICH TUN? S
O.K.
WAS SOLL ICH TUN? W
O.K.
WAS SOLL ICH TUN? UNTERSUCHE HOEHLE
ICH HABE EINEN BAEREN AUFGESCHRECKT.
WAS SOLL ICH TUN? N
```

Viele Spieler, soweit es sich um Anfänger handelt sogar die meisten, werden sich auf die gemachten Angaben verlassen und die Höhle niemals betreten. Denn dadurch, daß eine Untersuchung derselben, wie auch andere Aktionen, von der augenblicklichen Position möglich sind, wird jeder Verdacht auf eine großangelegte Höhlenwelt sofort zerstreut.

Erst eine Eingabe wie ´Betrete Höhle´ führt zu neuen Erkenntnissen:

```
130 AV=7

209 DATA BETRETE

2200 ON VN GOTO 5000,6000,7000,8000,9000
,10000,13000

13010 IF N=13 AND SP=5 THEN SP=12: PRINT
 "O.K.":GOTO 1080
```

ACHTUNG: Bevor Sie sich von der Funktion dieser Zeilen überzeugen, ist es erforderlich alle weiteren Räume einzugeben, wie auch die Verbindungen der bisher vorhandenen zu ändern (Zeilen 500 - 700).
Ebenso muß Zeile 110 an die neue Situation angepaßt werden.

Das schöne an dieser Realisation ist, daß der Zugang nicht einmal getarnt wird, im Gegensatz zu einer anderen Art der Ausführung, welche sich gleichfalls, zumindest seitens der Adventureproduzenten, großer Beliebtheit erfreut.

Aufgabe des Spielers ist es nun nicht mehr, nur einen versteckten Raum aufzuspüren und diesen mittels richtig

gewählter Worte zu betreten, sondern er muß in der Lage sein, einen Plan zu entwickeln, wie ein begehbarer Weg dorthin konstruiert werden kann.

Im einfachsten Fall handelt es sich um eine Tür, die zunächst verschlossen ist. Eine entsprechende Anzeige durch den Treiber wird sowohl durch die Objektbeschreibungen wie auch die Daten der für diesen Raum geltenden Zeile der Richtungstabelle gewährleistet:

```
xxxx DATA EINE TUER, TUER, 1

          REM RAUM 1
xxxx DATA IN EINER ZELLE.,0,0,0,0,0,0
```

Dieses übliche Verfahren sieht nun vor, daß der Spieler bei einer Untersuchung der Zelle die Tür, mit einem massivem Schloß versehen, entdeckt.

´Oeffne Tuer´ bzw. ´Oeffne Schloß´ weisen auf den fehlenden Schlüssel hin, wodurch der Spieler sich veranlaßt fühlt, alle weiteren Einrichtungsgegenstände des Raumes auf ihre Tauglichkeit zur Öffnung des Schlosses hin zu überprüfen.

Bei den meisten Adventures reicht die bloße Anwesenheit des Schlüssels im Inventory aus, die übrigen verlangen auch noch eine zweckentsprechende Benutzung des Schlüssels, eine Eigenschaft, die wir uns nun dank unserer Praxis erklären können.
Während die einen sich mit einem einfachen Test begnügen (OB(N)=-1), arbeiten die anderen, aufwendigeren Programme zusätzlich mit Flags.

Wie auch immer, steht die Tür erst offen, taucht eine entsprechende Richtungsangabe innerhalb der Kopfzeile auf, worauf wir auf eine Manipulation der für diesen Raum zuständigen Zeile des Feldes DURCHGANG (,) schließen.

```
xxxx IF N=y AND OB(z)=-1 AND SP=r THEN P
RINT "O.K.": DU(r,ri)=nr : GOTO 1080

y = Objektnummer Tür
z = Objektnummer Schlüssel
r = augenblicklicher Raum
ri= betreffende Himmelsrichtung
nr= neu zu erreichender Raum
```

Obige Zeile, eingesetzt in den für die Behandlung des Verbes ´Oeffne´ vorgesehenen Programmteil, vergewissert sich davon, daß der Spieler im Besitz des richtigen Schlüssels (z) ist und vor der richtigen Tür steht (r). Anschließend werden die Angaben in der Richtungstabelle entsprechend geändert.

LIMITIERUNGEN UND ZÄHLER

halte ich für besonders geeignet, um ein Adventure aus der Masse der übrigen hervorzuheben. Vermitteln doch besonders sie jedem Abenteuerprogramm die Faszination der Realität, und gestalten sie einige Abschnitte des Adventures zu wahren Denksportaufgaben.

Ihr Aufgabenbereich ist dabei weniger in einer aktiven Spielbeteiligung zu sehen, als vielmehr in einer Hintergrundkontrolle der Rahmenbedingungen, in die das Spiel eingebettet wird.

An erster Stelle muß hier wohl die Einschränkung des Inventories genannt werden, da ein Gewichtslimit inzwischen zum guten Ton dieser Computerspiele gehört.
Verständlich - erstens wirkt es unglaubhaft, wenn ein einzelner Mensch, wir gehen davon aus, daß es sich nicht um ein Supermann - Adventure handelt, beliebig viele Objekte

mit sich herumschleppen kann, und zweitens kann das Spiel auch zu einfach werden, wenn stets alle Handlungsgegenstände sogleich zur Verfügung stehen.

Technisch gesehen überwacht ein Zähler die Anzahl der Gepäckstücke, die der Spieler mit sich führt, und ist die vom Programmierer vorgesehene Tragkraft der Hauptperson ausgelastet, wird die Aufnahme weiterer Gegenstände verweigert.

Es muß also vor Beginn aller Handlungen die maximal erlaubte Stückzahl festgelegt werden:

```
185 IMAX=5
```

Die Inventory - Routine lassen wir ansonsten wie sie ist, denn es ist der Vorgang des Nehmens, dem wir erhöhte Aufmerksamkeit schenken müssen.
Allen Aktionen mit dem Verb ´Nimm´ wird nun eine weitere gemeinsame Voraussetzung zugrunde gelegt, weshalb sich zum zweiten Male eine Änderung der Zeile 6000 anbietet.
Leider sieht diese Planung nicht nur eine Umstrukturierung des betreffenden Blockes vor, sondern sie macht auch unseren Traum eines universell einsetzbaren Treibers mit allen Funktionen und Restriktionen zunichte.

Aus diesem Grunde ändern wir die Sprungtabelle in Zeile 2200 und führen den erforderlichen Test vor Beginn der eigentlichen Befehlsausführung durch:

```
2200 ON VN GOTO 5000,2210,7000,8000,9000
,10000
2201 REM TEST OB PLATZ IM INV
2210 ANZAHL=0
2220 FOR I=1 TO AO
2230 IF OB(I)=-1 THEN ANZAHL=ANZAHL+1
2240 IF ANZAHL=IMAX THEN PRINT"NEIN DANK
```

```
E. - ICH TRAGE SCHON GENUG !":GOTO 1080
2250 NEXT I
2260 GOTO 6000 : REM AUSFUEHRUNG NIMM --
2270 REM ------------------ ENDE INVTEST
```

Während diesem Zähler eine passive Rolle zukommt, und die Routine nur im Rahmen einer genau definierten Aktion aufgerufen wird, beschneiden andere Hintergrundkontrollen den Spieler nicht nur in der Freiheit seiner Handlungen, sondern können sogar zum Spielende führen.

Diese Behauptung gilt insbesondere für die folgenden Programmzeilen, deren Einbau in unser System dem Tüpfelchen auf dem ´i´ entspricht.

AVAILABLE LIGHT

ist die gebräuchliche Bezeichnung für eine bei einigen Adventureprogrammen anzutreffende Eigenart, die mit zusätzlichen Schwierigkeiten die Spieler zur Verzweiflung treiben kann.

Für Adventurespieler aus Passion scheinen die üblichen Hindernisse nicht ausreichend zu sein, und da bekanntermaßen nichts schwierieger ist, als das Leben in der Alltagswelt, muß zur Beseitigung dieses untragbaren Zustandes nichts weiter getan werden, als eben diese Welt noch perfekter nachzuahmen.

Ohne ´verfügbares Licht´ stehen wir im Dunkeln und werden nur schwerlich etwas sehen, geschweige denn präzise handeln können. Kommt dann noch die Tatsache hinzu, daß wir uns nicht in unserem Schlafgemach, wo dieser Zustand weitaus weniger hinderlich wäre, sondern in einer naturbelassenen, wilden und zerklüfteten Felsenlandschaft befinden, wird jeder weitere Schritt zu einem tödlichen Risiko.

Das kommt uns Adventureproduzenten natürlich sehr gelegen, und beim Einbau der nächsten Zeilen in unsere Adventures bereitet uns die Vorstellung der Reaktionen unserer Freunde, denen wir unser Werk selbstverständlich sofort nach Fertigstellung präsentieren wollen, eine zusätzliche Freude.

Glücklicherweise eignet sich jedes Abenteuerprogramm für einen Einbau dieses Hindernisses. Oft stehen weiträumige Höhlen oder unterirdische Gangsysteme zur Verfügung, andere Programme spielen sich in einem Haus ab; alles Orte, die einer künstlichen Beleuchtung bedürfen. Sollte tatsächlich einmal die Situation eintreten, daß die ganze Handlung unter freiem Himmel abläuft, muß eben der Lauf der Sonne nachvollzogen werden.

Um einen Sonnenauf- und -untergang realisieren zu können, reicht der Einsatz eines weiteren Zählers.

Zunächst legen wir fest, wie lange der Tag (die Nacht) dauern soll, wobei wir uns der Eingaben des Spielers als Zeiteinheit bedienen.
Nehmen wir an, 50 Züge nach Sonnenaufgang beginnt jeweils die Nacht. Dieser Wert wird ebenso typisch für jedes einzelne Adventure sein, wie die Anzahl der erreichbaren Punkte, weshalb die entsprechende Variable LM (Lichtmenge) gleichfalls bei Programmstart initialisiert wird:

```
186 LM=50 : LICHT=-1
```

Ihr nun schon recht umfangreiches Wissen zum Thema Adventure erlaubt es Ihnen, die Variable LICHT als Signalschalter zu interpretieren, und zwar in dem Sinne, daß eine -1 eine für alle Handlungen ausreichende Lichtmenge, 0 aber absolute Dunkelheit bedeutet.

Die beiden möglichen Stellungen dieses Schalters zeigen der Ausgaberoutine unseres Treibers, ob der Spieler etwas sehen kann, oder ob er im Dunkeln tappen muß:

```
1131 REM -------------------- KEIN LICHT
1132 IF LIGHT THEN 1140
1133 PRINT"ICH WEISS NICHT GENAU WO ICH
BIN."
1134 PRINT"ES IST ZU DUNKEL UM ETWAS ZU
SEHEN."
1135 PRINT: PRINT"AUCH DIE AUSGAENGE SEH
E ICH NICHT MEHR.":GOTO 1330
```

Nach Zeile 1132 werden die bislang benutzten Ausgaberoutinen abgearbeitet werden, falls Licht vorhanden ist.

Trifft das Gegenteil zu, informieren die Zeilen 1133 bis 1135 auf die gewohnte Art und Weise darüber, daß es nichts zu sehen gibt.
Mit dem Drucken der Trennungslinie kann der Programmverlauf durch die üblichen Zeilen fortgesetzt werden.

Ansonsten müssen wir nur noch für die jeweils richtige Stellung des Schalters LICHT sorgen, was nach Ablauf der mit LM festgelegten Anzahl von Spielzügen zu geschehen hat.

Der Zugzähler kann zu dieser Vergleichoperation nicht herangezogen werden, weil mit einem Tag- Nachtwechsel auch ein Rücksetzen des Zählers auf Null erforderlich wird; führen wir daher die Variable Lichtwechsel (LW) ein:

```
186 LM=50 : LIGHT=-1 : LW=0

1080 ZUG=ZUG+1 : LW=LW+1
1084 IF LW=LM THEN GOSUB 3000


2999 REM UNTERPROGRAMM LICHTSCHALTER
3000 IF LICHT=-1 THEN LICHT=0
3010 IF LICHT=0 THEN LICHT=-1
3020 LW=0
3030 RETURN
```

Zu Beginn einer neuen Eingaberunde wird überprüft, ob die vorgesehene Zeitdauer (LM) einer Periode erreicht ist. Wenn ja, wird das Unterprogramm Schalter aufgerufen und der Zustand von LICHT gewechselt. Mehr ist nicht erforderlich, den Abenteurer einem Tag und Nachtwechsel auszusetzen, was die Suche nach einer Lampe erforderlich machen kann.

Für Goldrausch wollen wir uns jedoch nicht mit der Simulation des Sonnenlaufes begnügen, sondern machen das Geschehen vom Besitz einer Lampe abhängig.

Dadurch stellen wir dem Spieler die zusätzliche Aufgabe, immer genügend Brennstoff für die Lichtquelle bereitzuhalten. In unserem Falle wird es sich dabei um Öl handeln, welches er in der Höhle des Bären finden kann. Dort füllt er es in die Flasche, die er zuvor wieder an sich genommen haben muß, und anschließend in die Lampe.
Jedes Füllen setzt dabei den Zähler Lichtwechsel wieder auf den Maximalwert.

Um die Sache zusätzlich zu komplizieren, wird die Laterne, zu dem Zeitpunkt, zu dem der Spieler sie findet, noch einen Rest Öl enthalten. Betritt er dann, ohne die Lampe nachzufüllen, die Tiefen der Mine, ist er unrettbar verloren. Ein weiterer Schritt, und der Abenteurer stolpert und bricht sich das Genick.

Da wir für das restliche Territorium keine Dunkelheit vorgesehen haben, besteht der erste Schritt zur Verwirklichung dieser Version in einer Änderung der Startwerte.
Eine Umschaltung erfolgt immer nur dann, wenn der Zähler Lichtwechsel gleich dem Inhalt von Lichtmenge ist.

Um dem Spieler zunächst jedoch eine fast unbegrenzte Anzahl von Spielzügen zu erlauben, legen wir LM mit Null und LW mit eins fest, und stellen dadurch sicher, daß LW nie gleich LM sein kann:

```
186 LM=0 : LICHT=-1 : LW=1 : L1=20
```

L1 kommt hinzu, weil die noch vorhandene Füllung geringer sein soll, als die später erfolgenden Nachfüllungen. Die übrigen Zeilen können vom vorangegangenen Beispiel übernommen werden.

Aktiviert wird diese Routine erst mit dem ersten Zünden der Lampe. Die Ausführung dieses Befehles wird neben der Ausgabe von ´O.K.´ auch die Zähler korrekt initialisieren:

```
11030 IF N=23 AND OB(23)=-1 THEN PRINT"O
.K. - DIE LAMPE BRENNT.":LM=L1:LW=1:GOTO
1080
```

Für die nächsten zwanzig Aktionen steht dem Spieler nun ausreichend Licht zur Verfügung. Mit dem Nachfüllen der Lampe kann er diesen Wert auf fünfzig vergrößern:

```
12010 IF N=23 AND OB(23)=-1 AND FL(6) TH
EN L1=50:LM=50:LW=1:PRINT"O.K.":GOTO1080
```

Um ein Nachfüllen des Brennstoffvorrates sowohl bei brennender als auch bei gelöschter Flamme zu ermöglichen, weisen wir der Variablen L1, wie auch LM, die Anzahl der nun durchführbaren Spielzüge zu.

Für den Fall, daß der Spieler sich an die Oberfläche begibt oder aus anderen Gründen die Lampe löscht, müssen wir den augenblicklichen ´Tankinhalt´ retten:

```
210 DATA LOESCHE
14000 IF N=23 AND OB(N)=-1 THEN L1=LM-LW
:PRINT "O.K.":GOTO1080
```

Ebenso gerettet werden müssen diese Werte mit dem Abspeichern eines Spieles:

```
1625 PRINT#2,SP:PRINT#2,L1:PRINT#2,LM:PR
INT#2,LW

1725 INPUT#2,SP:INPUT#2,L1:INPUT#2,LM:IN
PUT#2,LW
```

Damit haben wir sichergestellt, daß die Sache mit dem Licht auch wirklich ihren vorgezeichneten Verlauf nehmen muß.

Um dem Spieler aber die plötzliche, unangenehme Überraschung des unabwendbaren Endes zu ersparen, denn sollte er sich in den Tiefen der Erde befinden, während ihm das Öl ausgeht, wird er seine Haut nicht mehr retten können, ist eine Warnung fairerweise angebracht.

Folgende Zeile wird ihn kurz vor Brennschluß der Lampe von der drohenden Gefahr unterrichten:

```
1350 IF LM-LW<15 THEN PRINT"DAS LICHT DE
R LAMPE WIRD SCHWAECHER."
```

Ich hoffe, die vorangegangenen Beispiele werden Ihnen bei der Verwirklichung Ihrer eigenen Ideen helfen.
So wird es Ihnen sicherlich nicht schwerfallen, in Ihren Programmen beispielsweise dafür Sorge zu tragen, daß der Spieler rechtzeitig Nahrung zu sich nehmen muß, damit er weiterhin Dinge in die Hand nehmen und Aktionen durchführen kann.

Ebenso können Sie mittels eines weiteren Zählers Reaktionen von der verstrichenen Zeit abhängig machen.

ZUFÄLLE

Erdbeben, die eine Gesteinslawine auslösen, plötzlich auftauchende Monster, die natürlichen Feinde der Hauptperson im Adventure oder auch sich im unpassendem Moment öffnende Falltüren sind weitere, allseits beliebte Methoden, um den Spieler vom rechten Wege abzubringen.

Fast immer wird das Auftreten dieser Ereignisse vom Zufall gesteuert und das sogar in doppelter Hinsicht.

Ein Ungeheuer kann einen bestimmten Raum beleben und dem Spieler den Kopf abreißen:

```
1360 IF SP=10 THEN MS$(O)="DIESMAL HATTE
ICH KEINEN HONIG FUER DEN  BAER.":GOTO45
OO
```

Genausogut kann die Falle aber nur manchmal zuschnappen und wird zu anderen Zeiten dem Spieler kein Haar krümmen:

```
1370 IF SP=20 AND RND(1)>.8 THEN MS$(O)=
"DER BODEN BRACH UNTER MIR EIN.":GOTO450
O
```

Die dritte Version stellt eine Kombination der bereits vorgestellten Elemente dar.
Grundsätzlich wird man auch von einer Beweglichkeit der Feinde des Abenteurers ausgehen können, weshalb auch dem Monster ein gewisser Abschnitt der Adventurewelt als Spielwiese zur Verfügung gestellt werden sollte:

```
1380 IF (SP=13 OR SP=14 OR SP=16) AND RN
D(1)>.8THEN MS$(O)="WIEDER DER BAER.":GO
TO4500
```

VOM TEXT- ZUM GRAFIKADVENTURE

Neu vorgestellte Adventureprogramme weisen neben den besprochenen Features meist auch eine hervorragende optische Präsentation auf. Für jeden Raum des Adventures existiert ein Bild auf der Diskette welches bei Bedarf in den Bildschirmspeicher des Computers geladen wird.

Solch aufwendige Grafiken zu erstellen und in unser Adventuresystem einzubinden, wird uns ohne Hilfsmittel nicht möglich sein, dennoch soll Ihnen an dieser Stelle ein Weg vorgeschlagen werden, Ihre Programme attraktiver zu gestalten.

Auch Sie haben es vermutlich schon häufig bedauert, daß das Commodore 64 - Basic keine Befehle zur Erzeugung von Hires - Bildern enthält.
Daher haben Sie die Wahl des Einsatzes eines Hilfsprogrammes, wie beispielsweise der *SUPERGRAFIK* aus dem Hause DATA BECKER, oder Sie arbeiten mit den Grafikzeichen, die mittels der Tastatur eingegeben werden können.

Beide Lösungen erfordern allerdings einige kleine Änderungen unseres Treiberprogrammes, denn die Ausgaben werden nun an anderen Stellen des Bildschirmes gemacht werden müssen.

So werden auch Sie es vermutlich vorziehen, die Textausgabe auf den unteren Teil des Monitorbildes zu beschränken.
Dazu müssen Sie alle Zeilen des Treibers ändern die den Cursor positionieren (1070, 1080, 1090, 1130, 1390), und zwar in der Art, daß Sie die gewünschte neue Zeilennummer in die Speicherstelle 214 poken.
Bei Verwendung der Supergrafik wählen Sie entsprechende Koordinaten für den TEXT - Befehl.

Den Aufbau der jeweiligen Grafik überlassen Sie einigen Unterprogrammen, die vom Treiber aus aufgerufen werden.

Allerdings gilt es zu beachten, daß diese Druckroutine nicht nach jeder Eingabe des Spielers aufgerufen werden muß, weshalb wir eine weitere Kontrollvariable einführen:

```
1143 IF ALT<>SP THEN PRINT CHR$(147): GO
SUB 30000
1145 ALT=SP
```

Nach der Zeichnung eines Raumes wird dessen Nummer an die Variable ALT übergeben. Bewegt der Spieler sich in einen anderen Raum, ist die Bedingung in Zeile 1143 erfüllt, der Bildschirm wird gelöscht und es wird Zeile 30000 aufgerufen, welche anhand der Raumnummer die richtige Zeichenroutine wählt:

```
30000 ON SP GOSUB 31000, 32000, 33000
30010 RETURN

31000 REM RAUM 1
31999 RETURN

32000 REM RAUM 2
31999 RETURN
```

Anschließend wird der Programmlauf mit der Analyse des Befehls fortgesetzt.

5. KAPITEL
- DER EDITOR -

Die vorangegangenen Kapitel haben Ihnen das nötige Rüstzeug in die Hand gegeben, um gute Adventures zu schreiben.
Dennoch, obwohl wir mit unserem System ein Konzept entwickelt haben, welches zur Entwicklung der verschiedensten Adventures nur die Programierung der Daten, Bedingungen und Aktionen des betreffenden Spieles erforderlich macht, ist weiterhin ein beträchtlicher Arbeitsaufwand vonnöten, bevor die Testphase mit dem ersten Spielversuch beginnen kann.

Eine Vereinfachung der Arbeit würde ein sogenannter Adventure - Generator bedeuten, der, nachdem alle notwendigen Daten eingegeben wurden, auf der Diskette ein fertiges, spielbereites Programm hinterläßt.

Diese Lösung erfordert allerdings immer noch, daß die Planung des Spieles vollständig abgeschlossen ist. Denn bevor der Generator mit der Arbeit beginnen kann, hat der Adventureproduzent eine Liste aller Räume, Objekte, Verben usw. zu erstellen und diese Daten dann in einer Mammutsitzung einzugeben.
Treten während eines Testlaufes des so erstellten Programmes Fehler hervor, müssen diese weiterhin auf die übliche Art beseitigt werden.

Weitaus wünschenswerter wäre daher eine Lösung, die jeden einzelnen Gedanken des Autors sofort fixiert und darüberhinaus eine sofortige Überprüfung und auch Besserung ermöglicht.
Dies kann jedoch nur geschehen, wenn nicht ein fertiges Programm erzeugt, sondern eine Datei entwickelt wird, die alle für einen Spielablauf notwendigen Daten enthält.

Die Auswertung dieser Daten fällt dann einem Interpreter zu, der die zur Ausführung der verschiedensten Aktionen notwendigen Routinen enthält.

Wie solch ein Interpreter und der dazugehörige Editor auszusehen haben, wollen wir Ihnen in diesem Kapitel des Buches zeigen.

DIE DATEI

Es wurde bereits deutlich gemacht, daß das eigentliche Adventure nunmehr nur noch durch eine Datei repräsentiert wird.
Diese Datei muß alle für ein bestimmtes Spiel notwendigen Angaben in einer genau definierten Anordnung enthalten, damit für den Interpreter die Voraussetzung geschaffen ist, diverse Adventures verschiedener Länge zu laden.

Nehmen wir eins unserer programmmäßig realisierten Adventures unter die Lupe, so zeigt sich, daß diese bereits zu einem großen Teil interpretierend arbeiten.

Die Verben, wie auch die Objekte, werden nicht direkt zur Ausführung einer Aktion herangezogen, sondern es werden zunächst zwei Nummern ermittelt, deren Kombination genau festlegt, welche Eingabe der Spieler gemacht hat.

Als Voraussetzung für diese Arbeitsweise war es erforderlich, die Worte in Variablen einzulesen, ein Vorgang, der statt der Data - Zeilen auch direkt die Datei auf einer Diskette (Kassette) benutzen kann.

Auch die spieltypischen Daten haben wir bereits ermittelt und in den Zeilen 100 bis 200 als Kenndaten des Adventures in das Programm eingebracht.

Überlegen wir uns nun, welche Werte aus diesem Block für einen Interpreter notwendig sind, der zwar nicht alle Extras

unseres System beinhaltet, aber eine ausreichende Anzahl von Funktionen bietet, um ein Spiel möglich zu machen.

Zunächst finden wir einige Informationen über die Anzahl der vorhandenen Räume, Gegenstände und Verben.
Diese Grenzwerte erwiesen sich zur korrekten Steuerung der einlesenden Schleifen als notwendig und werden es aus gleichem Grunde auch für den Interpreter sein.
Ebenso werden wir die genaue Anzahl aller Mitteilungen festlegen müssen, denn sie werden fortan ohne Ausnahme in einem Variablenfeld aufbewahrt, und beizeiten ausgedruckt werden.

Gleiches gilt für die Bedingungen und Aktionen. In einem Feld, ähnlich der Richtungstabelle DU(,) werden wir diese Spieldaten zum Abruf bereithalten, weshalb wir die Variablen AC und BC (Aktionscodes und Bedingungscodes) zusätzlich in die Liste der Kenndaten aufnehmen.

Auf die Anzahl der Flags dürfen wir ebenfalls nicht verzichten, denn auch unserem Interpreter soll ein Abspeichern des Spielstandes möglich sein.

Als weiteren, für das Spiel wichtigen Wert müssen wir noch die Startposition des Spielers in unsere Datei aufnehmen.
Die Funktion des Systems wäre zwar auch gewährleistet, wenn SP nicht bei Spielstart initialisiert werden würde, aber dann müßte jedes Adventure immer in Raum 1 beginnen.
Auf den ersten Blick wird daraus zwar kein Nachteil ersichtlich, aber stellen Sie sich vor, nach einer kleinen Umstellung der Handlung soll der Abenteurer an einem völlig anderen Ort mit dem Spiel beginnen.
Findet sich dann keine Variable SP, die entsprechend neu initialisiert werden kann, müssen Sie alle Raumbeschreibungen, der neuen Planung gemäß umschreiben.

Auf eine Punktwertung können wir zunächst verzichten. Damit aber das Ende des Spieles definiert werden kann, werden wir

eine entsprechende Aktion vorsehen, die beispielsweise die Flags oder die im Inventory befindlichen Objekte auswerten kann.

Eine Ergänzung, die nichts mit dem Spielablauf zu tun hat, aber dennoch sinnvoll ist, sollten wir ebenfalls nicht vergessen.
So empfiehlt es sich, in der Datei auch Informationen über den Namen des Autors, den Namen des Adventures und vor allem das Datum der letzten Bearbeitung bereitzusellen, damit Sie die Version 5 mit nur noch drei Fehlern von der Version 1 mit mehr als zehn Fehlern, unterscheiden können.

DER AUFBAU DES EDITORS

Die Aufgabe des Editors ist es nun, dem Anwender die Eingabe wie auch später erforderliche Änderungen dieser Daten, auf bequeme und überschaubare Art möglich zu machen. Dazu werden eine Reihe unterschiedlicher Programmteile notwendig sein, deren Aufruf durch ein Menü erfolgen soll.

Die Reihenfolge wird dabei in gewissem Maße vorgegeben sein, denn solange die Räume noch nicht vorhanden sind, können die Gegenstände auch nicht plaziert werden.

Im einzelnen könnte der Arbeitsablauf der Konstruktion eines Adventures mit dem Editor folgendermaßen aussehen:

```
ADVENTURE EDITOR   VER 1.0
------------------------------
      0 - DATEN
      1 - RAEUME EINGEBEN
      2 - OBJEKTE EINGEBEN
      3 - VERBEN EINGEBEN
      4 - STARTPOSITIONEN
      5 - RAEUME VERBINDEN
      6 - BEDINGUNGEN & AKTIONEN
      7 - MELDUNGEN EINGEBEN
      8 - DISKETTENZUGRIFF
      9 - PROGRAMMENDE
------------------------------
      BITTE WAEHLEN SIE
```

Die gewünschte Funktion wählt der Anwender durch Eingabe der entsprechenden Ziffer, weshalb wir im Anschluß an die für obiges Bild erforderlichen Ausgabebefehle eine Programmzeile mit den Sprungzielen vorsehen:

```
240 ON A GOTO 1100,1200,1300,1400,1500,1
600,4000,5000
```

DIE EINGABEN

Bevor wir mit dem Aufbau der einzelnen Unterprogramme beginnen, hilft uns folgende Überlegung, den zu treibenden Aufwand möglichst gering und den Editor kurz zu halten.

Der Benutzer muß jeweils über die erwarteten Daten informiert werden, damit er korrekte Eingaben machen kann. Zweckmäßigerweise konstruieren wir einfache Eingabemasken, die entsprechende Hinweise enthalten.
Diese Masken werden, zumindest bei Eingabe der Räume, der Gegenstände und der Verben, weitgehend identisch sein, weshalb sich ein gemeinsamer Programmteil für diese Aufgaben anbietet.
Damit dieses Unterprogramm die richtigen Anweisungen geben kann, müssen diese Titel vor dem Aufruf zur Verfügung gestellt werden:

```
1100 PRINT CHR$(147):T1$="ORTE EINGEBEN
":T2$=" RAUM NR.:":T3$="ICH BIN "
1105 Z=AR
1110 GOSUB 510
```

Anschließend wird der Variablen Z die Anzahl der bisher vorhandenen Räume mitgeteilt und die Ein/Ausgabemaske aufgerufen (1110).

Zu ihrer Gestaltung muß der Cursor immer wieder an anderen Stellen positioniert werden, was in bereits dargelegter Weise geschieht.
Um das Programm jedoch nicht mit zahlreichen Poke - Anweisungen schwer lesbar zu machen, sehen wir ein weiteres Unterprogramm vor, daß den Cursor auf die gewünschte Zeile und auch Spalte setzt:

```
11000 POKE 211,SPALTE:POKE 214,ZEILE:SYS
 58732: RETURN
```

```
499 REM ------ UNTERPROGRAMM EIN/AUSGABE
500 PRINT CHR$(147);
510 ZE=0:SP=0:GOSUB 11000:PRINT T1$
520 Z=Z+1
555 ZE=0:SP=25:GOSUB 11000:PRINT T2$;Z
556 PRINT M3$
557 ZE=16:SP=0:GOSUB 11000:PRINT M4$
558 ZE=16:SP=0:GOSUB 11000:PRINT Z-1;:IF
A=1 THEN PRINT RAUM$(Z-1)
559 IF A=2 THEN PRINT OB$(Z-1)
560 IF A=3 THEN PRINT VERB$(Z-1)
563 ZE=20:SP=0:GOSUB 11000:PRINT T3$;:IN
PUT EINGABE$
564 IF LEFT$(EI$,1)="*"THEN GOTO 200
565 IF A=2 THEN ZE=22:SP=0:GOSUB 11000:I
NPUT"RUFNAME ";RN$(Z)
570 ZE=20:SP=0:GOSUB 11000:PRINT M4$
571 ZE=22:SP=0:GOSUB 11000:PRINT M4$
590 RETURN
591 REM -------------------- ENDE EIN/AUS
```

Zunächst werden die entsprechenden Hinweise ausgedruckt. In den Zeilen 558 bis 560 stellt eine Überprüfung des Wertes in A fest, um welche einzugebenden Daten es sich handelt, dann wird zur Erinnerung das letzte zuvor eingegebene Wort gedruckt.

Zeile 563 nimmt die Eingabe entgegen, die Zuweisung an die richtige Variable erfolgt innerhalb des rufenden Unterprogrammes (mit Ausnahme des zweiten Objektnamens, der in Zeile 565 eingegeben wird).
Die Zeilen 570 bis 571 löschen die Eingabezeilen und verhindern damit, daß bei der nächsten Eingabe Werte der vorherigen übernommen werden.

Die Beendigung der aufgerufenen Funktion erfolgt nach Eingabe eines Sternchens durch Zeile 564.

Analog zu diesem Beispiel wird die Eingabe der Objekte und Verben vorgenommen. Erst die Verbindung der Räume erfordert einen anderen Aufbau.

VERBINDUNGEN DER RÄUME

Zunächst müssen alle möglichen Räume angezeigt werden, damit der Anwender die richtigen Raumnummern eingeben kann. Da diese Funktion ebenfalls für die Plazierung der Objekte notwendig ist, erstellen wir ein weiteres Unterprogramm (12000 - 12070).

Daran anschließend werden für alle Räume der Reihe nach die sechs Richtungen abgefragt und die Eingaben direkt in die entsprechende Position der Richtungstabelle geschrieben:

```
1499 REM --------------- RAEUME VERBINDEN
1500 FOR R1=I2 TO AR
1520 GOSUB 12000
1530 PRINT "RAUM";R1;"FUEHRT IM NORDEN I
N RAUM";:INPUT DU(R1,1)
1540 PRINT "RAUM";R1;"FUEHRT IM SUEDEN I
N RAUM";:INPUT DU(R1,2)
```

```
1585 NEXT R1
1590 GOTO 200
1591 REM ------------- ENDE VERBINDUNGEN
```

AKTIONEN & BEDINGUNGEN

Bevor wir mit der Eingabe der wichtigsten Daten, den Bedingungen und Aktionen fortfahren, müssen wir ein System zur Codierung entwickeln.

Zur Aufbewahrung der Daten verwenden wir ein weiteres zweidimensionales Feld, wobei die Reihen diesmal durch die Verben, und die Spalten durch die Objekte vorgegeben werden.
Um die Bedingungen auch für uns interpretierbar zu machen, entscheiden wir uns für eine Codierung durch Buchstaben. Zahlen würden bei der Bearbeitung zwar einen Geschwindigkeitsvorteil bedeuten, aber die Bedeutung einer zwölfstelligen Zahl zu erkennen, ist mit Sicherheit schwieriger, als die Interpretation eines Strings wie "RF3S22" vorzunehmen, wenn dabei Konventionen, wie aus den folgenden Abbildungen ersichtlich, getroffen wurden.

```
BITTE ALLE BEDINGUNGEN FUER DIE AKTION
UNTERSUCHE TUER; EINGEBEN

      R   - OBJEKT IST IM RAUM
      I   - OBJEKT IST IM INVENTORY
      N   - OBJEKT IST NICHT VORHANDEN
      FX  - FLAG X IST GESETZT
      GX  - FLAG X IST GELOESCHT
      SXX - SPIELER IST IM RAUM XX

ALTER CODE ===>
BEDINGUNGEN? RS02
```

Diese Bedingungen sind völlig ausreichend, um Aktionen des Spielers nur zu den von der Handlung vorgeschriebenen Zeitpunkten durchführbar zu machen.

Auch die vorgesehenen Aktionen entsprechen im wesentlichen den auf Seite 66 gemachten Anmerkungen, neu hinzugekommen sind lediglich die Befehle Dxy und E.
Der Befehl D soll den Interpreter veranlassen, einen Ausgang aus dem augenblicklichen Raum nach Raum x in Richtung y sichtbar werden zu lassen, E beendet das Spiel und informiert den Spieler über seinen Sieg.

```
RS02 ERFUELLT, FOLGENDE AKTION:
-----------------------------------------
    V   - TUER VERSCHWINDET
    I   - TUER KOMMT INS INV
    NXX - OBJEKT XX ERSCHEINT NEU
    DXY - DURCHGANG N. RAUM X
    SXX - SPIELER NACH RAUM XX
    FX  - FLAG X SETZEN
    LX  - FLAG X LOESCHEN
    MXX - MELDUNG XX AUSGEBEN
    T   - SPIELER STIRBT
    E   - ENDE, DA GEWONNEN
-----------------------------------------
ALTER CODE ===>
AKTION? M09
```

DIE MITTEILUNGEN

können ohne weitere Umstände sofort bei der Eingabe in die richtigen Variablen geschrieben werden. Entsprechend kurz fällt der für diesen Zweck vorgesehene Programmteil aus:

```
3999 REM ------------------ MITTEILUNGEN
4000 PRINT"MITTEILUNGEN EINGEBEN"
4010 PRINT M3$
4020 AM=AM+1
4030 PRINT AM;INPUT MS$(AM)
4040 IF LEFT$(MS$(AM))="*" THEN AM=AM-1:
GOTO 200
4050 GOTO 4020
4060 REM ------------- ENDE MITTEILUNGEN
```

AM behält die Kontrolle über die Anzahl der eingegeben Nachrichten und muß nach Beendigung der Funktion durch Eingabe eines Sternchens vermindert werden.

DISKETTENZUGRIFFE

werden erforderlich, um die Datei zu speichern bzw. um sie zur erneuten Bearbeitung wieder in den Rechner zu laden. Damit die Datei, auch wenn ein Spiel noch nicht vollständig realisiert wurde, für den Interpreter bereits spielbar ist, müssen vor einem Abspeicherungsvorgang zusätzliche Daten eingegeben werden.

Wichtig für den Interpreter ist vor allen Dingen die Wortlänge, denn dieser soll den für Variablen vorgesehenen Speicherplatz nicht dadurch mehr als nötig belegen, daß er die vollständigen Wörter speichert, sondern er soll nur die wirklich benötigten Buchstaben einladen.

Bereits erwähnt wurde, daß der Startraum des Spielers in der Datei enthalten sein muß, wie auch die Anzahl der verwendeten Flags für den Interpreter wichtig wird, sobald die SAVE GAME Funktion aufgerufen wird.

Nach Eingabe dieser Werte werden alle Daten als sequentelles File auf der Diskette abgelegt. Cassettenbenutzer beachten bitte die Hinweise im vorangegangenen Kapitel.

Ansonsten weist das Listing des Adventure - Editors keine Besonderheiten auf, so daß Sie, nicht zuletzt auch wegen der Remarks, die auf die einzelnen Blöcke hinweisen, die Funktionsweise verstehen dürften und das Programm nach Ihren Wünschen erweitern können.

DER ADVENTURE INTERPRETER

Auch das Verständnis des Interpreters wird Ihnen kaum Schwierigkeiten bereiten, ist er doch ähnlich unserer Adventureprogramme aus drei Blöcken aufgebaut.

Im ersten Teil des Programmes werden wieder die Variablen mit den für das Spiel erforderlichen Daten initialisiert. Statt der DATA - Zeilen mit zugehörigen READ - Anweisungen benutzen wir nun ein mit dem Editor erstelltes Datenfile.

Nachdem Sie die gewünschte Datei gewählt haben, wird sie von den Zeilen 50 bis 200 ausgewertet und aufbereitet.

Der Programmblock von Zeile 1000 bis einschließlich Zeile 2160 entspricht dem von uns entwickeltem Treiber, Änderungen sind nicht erforderlich.

Völlig neu ist jedoch die Art der Ausführung jeglicher Aktionen.
Ein einziges Programm, der Interpreter, soll in der Lage sein, zahlreiche, jeweils voneinander verschiedene Spiele ablaufen zu lassen. Deshalb ist es nicht möglich, starre Strukturen, die immer nur auf eine einzige Situation zugeschnitten sind, zu verwenden. Aus einer Reihe von Einzelaktionen müssen die gerade erforderlichen herausgesucht und ausgeführt werden, bis die Summe ihrer Wirkungen das gleiche Resultat erzielt hat.

Diese Lösungsstrategie verfolgen wir sowohl bei der Überprüfung der Bedingungen als auch zur Ausführung der Handlungen.

```
2200 FOR AB=1 TO LEN (BC$(VN,N))
2210 BD$(AB)=MID$(BC$(VN,N),AB,1)
2220 NEXT AB
```

Zunächst wird der Endwert der Schleife ermittelt, der von der Anzahl der für die Codierung einer bestimmten Aktion benutzten Buchstaben abhängig ist. Anschließend erfolgt eine Zerlegung des gesamten Strings in die einzelnen Bedingungen, die jeweils einem Element der Liste BD$() zugewiesen werden.

Eine weitere Schleife überprüft nun die Erfüllung jeder einzelnen Voraussetzung innerhalb einer weiteren Schleife.

Jedoch war es bei der Codierung erforderlich, für einige Bedingungen zwei und mehr Parameter festzulegen. So erfordert der Code S (Anwesenheit des Spielers in einem bestimmten Raum) die Angabe einer Raumnummer, wie auch das betreffende Flag bei F und G genannt werden muß.
Aus diesen Gründen verzichten wir auf eine For/Next - Schleife und wählen stattdessen einen Aufbau, der es uns

gestattet, den Schleifenzähler um beliebige, differierende Werte zu erhöhen.

Neben dieser Variablen (X) benutzen wir eine weitere Variable ERGEBNIS, die nach jedem Schleifendurchgang logisch wahr (-1) sein soll, wenn die Bedingung erfüllt war und die nach Verlassen der Schleife logisch nein (0) sein wird, wenn nur eine einzige Forderung nicht erfüllt ist.

```
2300 X=0:ER=0
```

Jeder Schleifendurchgang soll mit einer Erhöhung des Zählers beginnen, sofort danach wird geprüft, ob der Endwert, der ja von der Anzahl der Bedingungen abhängig ist, erreicht ist oder nicht.

```
2310 X=X+1
2320 IF X=AB+1 THEN 2500 : REM ALLE BEDI
NGUNGEN UEBERPRUEFT
```

Abhängig von ER wird nun entschieden, ob eine Reaktion stattfindet, oder ob der Spieler weitere vorbereitende Handlungen durchführen muß:

```
2500 IF NOT ER THEN PRINT"DAS GEHT IM MO
MENT NICHT.":GOTO 1080
3999 REM ---- ALLE BEDINGUNGEN ERFUELLT
4000 PRINT "O.K."
```

Innerhalb des Schleifenrumpfes werden die Bedingungen der Reihe nach überprüft, wobei die betreffende Programmzeile auf die gleiche Art ermittelt wird, wie beispielsweise die Routinen zur Behandlung der Ein - Wort Befehle innerhalb unseres Treiberprogrammes:

```
2330 IF BD$(X)<>"R"THEN 2250
2340 IF OB(N)=SP THEN ER=-1 : GOTO 2310
2350 IF BD$(X)<>"I"THEN 2270
```

```
2360 IF OB(N)=-1 THEN ER=-1 : GOTO 2310
2370 IF BD$(X)<>"N"THEN 2390
2380 IF (OB(N)<>SP AND OB(N)<>-1) THEN
ER=-1 : GOTO 2310
2390 IF BD$(X)<>"S"THEN 2410
2400 R1$=BD$(X+1)+BD$(X+2):X=X+2:IF SP=V
VAL(R1$) THEN ER=-1 : GOTO 2310
2410 IF BD$(X)<>"F" THEN 2450
2420 IF FL(VAL(BD$(X+1))) THEN 2440
2430 X=X+1:GOTO 2450
2440 ER=-1:X=X+1:GOTO 2310
2450 IF BD$(X)<>"G" THEN 2310
2460 IF NOT FL(VAL(BD$(X+1))) THEN 2480
2470 X=X+1:GOTO2310
2480 ER=-1:X=X+1:GOTO 2310
2490 GOTO 2310
```

Außer den Zeilen 2400 bis 2470 werden auch diese Zeilen Sie vor keinerlei Probleme stellen.
In Zeile 2400 werden auch noch die nächsten zwei Elemente des Bedingungscodes BD$ herangezogen, um die Raumnummer zu ermitteln, anschließend wird der Zähler aktualisiert und eventuell die Ergebnisvariable auf wahr gesetzt.
Ähnlich arbeiten die für die Auswertung der Flags vorgesehenen Zeilen.

Identisch ist die Arbeitsweise der einen Befehl ausführenden Programmzeilen.
Innerhalb dieses Blockes wäre nur auf die Zeilen 5200 bis 5290 hinzuweisen, die es ermöglichen, beispielsweise eine Tür zu öffnen. Durch ihre Manipulationen werden die korrekten Werte in die Richtungstabelle geschrieben, so daß einerseits der neue Durchgang sichtbar wird (5220), andererseits aber auch ein Rückweg vorhanden ist, wenn der Spieler sich in den neuen Raum begeben hat (5230 bis 5290).

Die Zeilen von 5500 bis 5700 enthalten jeweils eine kurzes Unterprogramm, welche der Ermittlung einer zweistelligen Zahl (5500), einer einzigen Ziffer (5600) beziehungsweise zweier Ziffern dienen (5700).
Benötigt werden diese Werte zur Ausgabe der richtigen Mitteilungen, der korrekten Behandlung der Flags wie auch für die Öffnung eines Durchganges zu einem bestimmten in eine der sechs Richtungen.

Das vollständige Listing des Interpreters entnehmen Sie bitte ebenfalls dem nächsten Kapitel.

6. KAPITEL
- ADVENTUREPRAXIS -

Dieses Kapitel wird Ihnen weniger zeigen, wie man´s macht, sondern was man machen kann.

Sie werden die vollständigen Listings zweier Adventures finden, die unter Verwendung der in diesem Buch entwickelten Programmteile und Routinen geschrieben wurden, wie auch Bedienungsanweisungen für die im folgenden abgedruckte Software.

Zum einen handelt es sich dabei um Goldrausch, ein Adventure, zu dem schon fast zuviel gesagt wurde, zum anderen um das ´Verzauberte Schloß´, ein Abenteuerprogramm, zu dessen Lösung wir Ihnen keinerlei Hinweise geben werden.

Leider ist es jedoch so, daß Sie, wenn Sie dieses Buch durchgearbeitet haben, die Programme fast wie eine Anweisung zur Lösung lesen werden können, weshalb es empfehlenswert wäre, die Arbeit des Eintippens mit jemandem zu teilen.

Dies gilt insbesondere für die Programmzeilen ab Nummer 5000, da hier das Spielgeschehen programmiert ist.

Im Anschluß daran finden Sie zwei kürzere Programme, bei denen es sich um einen Adventure - Editor und um den dazugehörigen Interpreter handelt.
Beide Programme zusammen erlauben es Ihnen, menügesteuert Adventures zu erzeugen und diese bereits vom ersten Schritt der Entwicklung an zu spielen.

SPIELANLEITUNG ADVENTURES

Nachdem Sie das Programm mit RUN gestartet haben, erscheinen nach dem Titelbild und weiteren, allgemeinen Spielhinweisen die ersten Beschreibungen und Mitteilungen auf dem Monitor.
Sie müssen zwischen der oberen und unteren Bildschirmhälfte unterscheiden: - in der oberen Hälfte finden Sie eine Beschreibung des Ortes, an dem Sie sich gerade befinden, direkt darunter werden Ihnen alle sichtbaren Gegenstände aufgezählt und in einer weiteren Zeile werden die Richtungen genannt, in die Sie sich bewegen können.
Die untere Hälfte dient Ihnen dazu, Ihre Befehle an die Spielfigur einzugeben, ebenso werden dort die von der Durchführung Ihrer Eingabe abhängigen Informationen an Sie ausgegeben.

IHRE EINGABEN:

Ihre üblichen Eingaben werden meist aus zwei Worten, einem Verb und einem Objekt, bestehen. Achten Sie darauf, Bezeichnungen zu verwenden, wie sie auch im oberen Teil des Bildschirmes ausgegeben werden.
Das bedeutet allerdings nicht, daß Sie auf eine Mitteilung wie:

ICH SEHE EINE(N) ROTE TÜR.

WAS SOLL ICH TUN:

mit OEFFNE ROTE TUER reagieren, meistens wird die Eingabe OEFFNE TUER völlig ausreichend sein.

Zusätzlich werden Ihnen einige weitere Eingaben erlaubt sein, die sogenannten Ein - Wort Befehle, die bestimmte Funktionen erlauben, welche für das Spiel nicht unbedingt erforderlich sind, es aber vereinfachen.

RÄUMLICHE FORTBEWEGUNG:

Sie können sich generell immer in die angegebenen Richtungen bewegen. Dazu reicht es aus, den Anfangsbuchstaben der jeweiligen Richtung einzugeben (außer für Bewegungen nach oben, hier müssen Sie zur Unterscheidung von Osten OB eingeben): Norden, Süden, Osten, Westen : N, S, O, W; Oben, Unten : OB, U.

Es gibt jedoch auch Ausnahmen und manchmal kann eine Eingabe wie BETRETE ... nutzbringend sein.

MANIPULATION VON OBJEKTEN:

Natürlich wollen Sie in der imaginären Welt nicht nur auf eine Besichtigungsreise gehen, sondern Sie wollen handeln und eine Aufgabe lösen.
Dazu müssen Sie die Gegenstände dieser Welt erst einmal entdecken, dann können Sie sie UNTERSUCHEn, NEHMEN, BENUTZEn usw. (Wenn Sie einmal nicht weiter wissen, sollten Sie sich an diese Worte erinnern !).

Nehmen wir an, Sie finden irgendwo einen Schlüssel. Sie denken ´prima, doch was soll ich damit ?´
Im wirklichen Leben würden Sie ihn sich ansehen, würden ihn genau untersuchen und dann entscheiden, ob Sie ihn liegenlassen oder ob Sie ihn für brauchbar oder sogar wertvoll halten und daher mitnehmen. Diesen logischen Handlungsablauf sollten Sie im Adventure nachvollziehen.

Geben Sie ein: UNTERSUCHE SCHLÜSSEL. Als Antwort erhalten Sie vielleicht: ES IST DER SCHLÜSSEL ZU MEINER WOHNUNG oder: ES IST EIN AUTOSCHLÜSSEL oder: ER IST AUS PUREM GOLD. Natürlich würden Sie keinen dieser Schlüssel liegen lassen, sondern alle mitnehmen, geben Sie also ein: NIMM SCHLÜSSEL. Auf dem Bildschirm erscheint als Mitteilung ein O.K. (Dieses O.K. werden Sie oft sehen, es bedeutet, daß der

Adventureinterpreter Ihre Eingabe verstanden und ausgeführt hat.). Weiterhin wird der Schlüssel in der Kopfzeile ICH SEHE nicht mehr erscheinen, denn da Sie ihn eingesteckt haben, befindet er sich jetzt in Ihrer Manteltasche (oder ähnlichem), jedoch nicht mehr im Raum.

Sollten sich dann im Laufe eines Spieles zahlreiche Gegenstände in Ihren Taschen angehäuft haben, werden Sie wahrscheinlich die Übersicht verlieren, deshalb bietet Ihnen der Interpreter die Möglichkeit,

INVENTUR zu machen. Immer wenn Sie wissen wollen, ob Sie beispielsweise irgendwelche Sachen mit sich herumtragen, die geeignet wären, eine bestimmte Aufgabe zu erfüllen, so geben Sie einfach INV ein.
Als Antwort erscheint sofort: *ICH TRAGE FOLGENDES MIT MIR:* sowie eine Liste all dieser Gegenstände.

SPIELUNTERBRECHUNG:

Sie werden es wohl kaum schaffen, ein Adventure in einem Anlauf zu lösen, und natürlich wollen Sie am nächsten Tag nicht wieder von vorne beginnen.
Daher bieten Ihnen diese Spiele die Möglichkeit, den augenblicklichen Spielstand abzuspeichern.

Wenn Sie ein Spiel vorläufig beenden wollen, so geben Sie einfach *SAVE* ein. Der Interpreter fragt Sie dann nach einem Namen, unter dem er die augenblickliche Situation abspeichern soll.

Dieses Abspeichern empfiehlt sich auch zwischendurch: So könnte beispielsweise die Situation auftreten, daß Sie mit falschem Schuhwerk eine steile Felswand besteigen wollen. Hier ist die Wahrscheinlichkeit sehr groß, daß Sie abstürzen und sich Ihr Genick brechen; als Folge davon müßten Sie von vorne beginnen und alle Eingaben

wiederholen, eine uninteressante Tätigkeit, vor der Sie ein rechtzeitiges
Abspeichern bewahrt hätte.

FORTSETZUNG EINES UNTERBROCHENEN SPIELS:

Haben Sie den Spielstand gespeichert, geben Sie einfach *LOAD* ein. Der Interpreter fragt Sie dann, unter welchem Namen Sie den Spielstand abgespeichert hatten und setzt nach Drücken der <RETURN> - Taste das Spiel an der unterbrochenen Stelle fort.

SONSTIGE HINWEISE:

Wenn Sie ein Adventure spielen, müssen Sie zunächst herausfinden, welcher Art das Adventure ist. Entweder haben Sie eine Aufgabe zu lösen, einen Ausgang zu suchen oder Schätze zu sammeln. Sie sollten zunächst alles UNTERSUCHEn und sich möglichst logisch verhalten (so brauchen Sie zum ÖFFNEn einer VERSCHLOSSENEn TÜR zunächst einen SCHLÜSSEL, eine BRECHSTANGE o. ä.). Falls Sie sich auf dem richtigen Weg befinden, gibt der Interpreter Ihnen das durch eine Meldung der Art DAS GEHT IM MOMENT NICHT oder ICH VERSTEHE NICHT, WAS DU MEINST zu verstehen.
Diese Mitteilungen bedeuteten, daß die Worte Ihrer Eingabe verstanden werden, zur Durchführung der Aktion aber noch nicht alle Bedingungen erfüllt sind (z. B. fehlt der SCHLÜSSEL).

Wird ein Verb oder das Objekt nicht verstanden, so wird Ihnen darüber Mitteilung gemacht, und Sie sollten versuchen, das gleiche Ziel mit anderen Worten zu erreichen.

O.K. signalisiert Ihnen, daß eine Eingabe verstanden wurde und daß eine Reaktion stattgefunden hat. Meist erscheinen

weitere Mitteilungen in der unteren Bildschirmhälfte, welche diese Reaktion genauer beschreiben.

TIPS ZUR LÖSUNG EINES ADVENTURES

Wenn Sie in verfahrenen Situationen überhaupt nicht mehr weiter wissen, können Sie es zunächst mit HELP versuchen.

Gelangen Sie auf diese Weise nicht in den Besitz weiterer Informationen, rufen Sie sich alle bisher entdeckten Gegenstände in die Erinnerung zurück.
Mit ziemlicher Sicherheit kommt jedem Objekt eine Aufgabe zu; es nur zur Dekoration eines Raumes einzusetzen, dazu wäre der Programmieraufwand zu groß.

Hilfreich wäre dazu eine Liste aller vom Programm verstandenen Worte, die Ihnen möglicherweise sogar vom Adventure selbst auf den Befehl VOKabeln / VOCabulary hin erstellt wird.

Als ebenso nützlich erweist sich eine Karte, auf der Sie alle Räume mit Ihren Verbindungen eintragen, wobei Sie natürlich auch die Fundorte der Gegenstände nicht vergessen.

Im Falle des Aufenthaltes in einem Labyrinth ist die Erstellung einer solchen Karte jedoch gar nicht so einfach, wie Sie es sich jetzt vielleicht vorstellen.

Statt wirklich vorhandener Räume hat man oft die dreifache Anzahl kartographiert. Der einzige Trick, um dies zu verhindern, besteht darin, in jedem Raum des Irrgartens einen Gegenstand aus dem Inventory abzulegen und die einzelnen Aufenthaltsorte damit unverwechselbar zu machen.

Und als letzten Hinweis noch einmal:
Speichern Sie Ihr Spiel zwischendurch immer wieder ab !

```
1 REM --- GOLDRAUSCH, VERSION 1.0 --
2 REM      (C) 1984 BY WALKOWIAK
10 REM ----------------------- TITELBILD
11 POKE53281,11:POKE53280,12:PRINTCHR$(1
42)
12 PRINT"
13 PRINT"
14 PRINT"
15 PRINT"
16 PRINT"
17 PRINT"
18 PRINT"          (C) 1984
19 PRINT"               BY
20 PRINT"           JOERG WALKOWIAK
21 PRINT"
22 PRINT"
23 PRINT"
24 PRINT"
25 PRINT"
26 PRINT"
27 PRINT"
28 PRINT"
```

```
29 PRINT"        |              |  /──────
  ▆   |
30 PRINT"        |              | /
/  ▆└──┘
31 PRINT"        └──────────────┘/       /
32 PRINT"                   /────────────/
33 PRINT"  ▆     AUS DEM DATA BECKER BUC
H
34 PRINT"▆                 ADVENTURES -▆  /
35 PRINT"▆      UND WIE MAN SIE PROGRAMMI
ERT
36 PRINT"▆ -           /──────────/";
99 FOR I=1 TO 10000:NEXT
100 REM ----------------------- KENNDATEN
101 :
110 AR=31
120 AO=44
130 AV=11
140 WL=4 : AM=10
150 SPIELER=1
160 AF=7
170 WMAX=60
171 WERTUNG=0
180 ZUG=0
185 IMAX=4 : REM MAXIMAL ZU TRAGENDE OBS
186 LM=0 :LICHT=-1:LW=1:L1=20
190 DIMRA$(AR),DURCHGANG(AR,6),OB$(AO),
RN$(AO), OB(AO),FL(AF),MS$(AM),VE$(AV)
199 :
200 REM ------------------------- VERBEN
201 DATA UNTERSUCHE
202 DATA NIMM
203 DATA "LEG "
204 DATA OEFFNE
205 DATA BENUTZE
206 DATA ZERSTOERE
207 DATA ZUENDE
208 DATA FUELLE
209 DATA BETRETE
210 DATA LOESCHE
211 DATA BEFESTIGE
```

```
299 :
300 REM -------------------- GEGENSTAENDE
209
301 DATA"VIELE GROSSE BAEUME","BAEUME",1
302 DATA"VIELE GROSSE BAEUME","BAEUME",2
303 DATA"EINIGE FELSBROCKEN","FELSEN",2
304 DATA"EINE VERFALLENE HOLZHUETTE","HU
ETTE",3
305 DATA"EINE VERSCHMUTZTE KORBFLASCHE",
"FLASCHE",0
306 DATA"HONIG","HONIG",0
307 DATA"EINE HOLZKISTE","KISTE",3
308 DATA"EIN KLAPPRIGES REGAL","REGAL",0
309 DATA"ETWAS SPRENGSTOFF","SPRENGSTOFF
",0
310 DATA"EIN DUESTERES ERDLOCH","ERDLOCH
",0
311 DATA"EINE ROSTIGE EISENTRUHE","TRUHE
",0
312 DATA"*SILBERMUENZEN*","SILBER",0
313 DATA"EINE DUESTERE FELSENHOEHLE","HO
EHLE",5
314 DATA"EINEN GRIMMIG DREINBLICKENDEN B
AEREN","BAER",0
315 DATA"ZAHLLOSE NIEDRIGE BUESCHE","BUE
SCHE",4
316 DATA"MEHRERE EISENSTANGEN","EISENSTA
NGE",6
317 DATA"*NUGGETS*","NUGGETS",5
318 DATA"EINE EISENSTANGE","EISENSTANGE"
,0
319 DATA EINE ALTE LORE,LORE,7
320 DATA VERROTTETE SCHIENENSTRAENGE,SCH
IENEN,7
321 DATA EIN SEIL,SEIL,0
322 DATA EINE SCHWERE SPITZHACKE,HACKE,8
323 DATA EINE ALTE LATERNE,LATERNE,8
324 DATA DAS GANGENDE,GANGENDE,9
325 DATA EISENHAKEN,HAKEN,0
326 DATA EINEN SCHACHT,SCHACHT,9
327 DATA EINE BRETTERWAND, BRETTERWAND,1
```

```
0
328 DATA ABBAUSCHUTT,SCHUTT,11
329 DATA ALTE SAECKE,SAECKE,0
330 DATA FEUCHTE STEINWAENDE,WAENDE,13
331 DATA EINE UEBELRIECHENDE FLUESSIGKEI
T,FLUESSIGKEIT,17
332 DATA EINE LEICHE,LEICHE,18
333 DATA *SILBERKLUMPEN*,KLUMPEN,0
334 DATA *GOLDMUENZEN*,MUENZEN,0
335 DATA SPINNWEBEN,SPINNWEBEN,20
336 DATA EINE GOLDENE WAND,WAND,22
337 DATA EIN SCHILD,SCHILD,22
338 DATA *GOLD*,GOLD,31
339 DATA EIN SEEUFER,UFER,30
340 DATA DEN SEE,SEE,30
341 DATA GOLDENE STEINE,STEINE,0
342 DATA ZUENDSCHNUR,ZUENDSCHNUR,0
343 DATA EIN LUNTERO, LUNTERO,-1
344 DATA -,PARADIES,0
499 :
500 REM ------------ RAUMBESCHREIBUNGEN
501 DATA"IM WALD.",1,1,1,2,0,0
502 DATA"IM WALD.",2,1,1,3,0,0
503 DATA"IM WALD VOR EINEM FELSHANG.",0,
4,2,0,0,0
504 DATA"AUF EINER WALDLICHTUNG.",3,0,5,
0,0,0
505 DATA"AUF EINER LICHTUNG AM RANDE
 EINES  BERGHANGES.",0,0,6,4,0,0
506 DATA"AM EINGANGSSTOLLEN EINES ALTEN
 BERGWERKES.",7,0,1,5,0,0
507 DATA IM EINGANGSSTOLLEN.,8,6,0,0,0,0
508 DATA IN EINER NISCHE DES GANGES.,9,7
,8,10,0,0
509 DATA AM ENDE DES GANGES.,0,8,0,0,0,0
510 DATA IN EINEM SEITENGANG.,11,0,8,0,0
,0
511 DATA AN EINER ALTEN ABBAUSTELLE.,0,1
0,0,0,0,0
512 DATA IN DER HOEHLE.,0,5,0,13,0,0
513 DATA IN DER HOEHLE.,0,0,12,14,0,0
```

```
514 DATA IN DER HOEHLE.,0,16,13,15,0,0
515 DATA IN DER HOEHLE.,0,0,14,0,0,0
516 DATA AUF EINEM GANG.,14,0,0,17,0,0
517 DATA IN EINER NEBENHOEHLE.,0,0,16,0,
0,0
518 DATA AUF DEM BODEN DES SCHACHTES.,0,
0,19,0,0,0
519 DATA AUF EINEM KURVENREICHEN GANG.,2
4,0,20,18,0,0
520 DATA IN EINEM BREITEN GANG.,0,0,21,1
9,0,0
521 DATA IN EINER ALTEN ABBAUSTRECKE,22,
11,11,20,11,0
522 DATA IN EINEM FELSENDOM.,0,21,0,0,27
,0
523 DATA IN EINEM UNTERIRDISCHEM PARADIE
S.,0,0,0,0,0,0
524 DATA AUF EINEM KURVENREICHEN GANG.,2
6,19,24,24,24,24
525 DATA AUF EINEM KURVENREICHEN GANG.,2
7,24,24,29,26,24
526 DATA AUF EINEM KURVENREICHEN GANG.,2
7,24,26,27,27,24
527 DATA AUF EINEM KURVENREICHEN GANG.,2
5,24,26,27,26,24
528 DATA AUF EINEM KURVENREICHEN GANG.,2
4,26,26,27,0,0
529 DATA AUF EINEM KURVENREICHEN GANG.,0
,27,30,27,0,0
530 DATA VOR EINEM UNTERIRDISCHEM SEE.,2
8,0,0,0,0,0
531 DATA IN EINER KLEINEN HOEHLE.,30,0,0
,0,0,0
599 :
600 REM ------------------- MITTEILUNGEN
601 MS$(1)="ICH SEHE NICHTS BESONDERES."
602 MS$(2)="SO STARK BIN ICH NICHT."
603 MS$(3)="WIE STELLST DU DIR DAS VOR ?
"
604 MS$(4)="DER BAER NIMMT DEN HONIG UND
"
```

```
605 MS$(5)="VERSCHWINDET IN DER TIEFE DE
R HOEHLE."
606 MS$(6)="WIE KANN ICH DIE KETTE ZERRE
ISSEN ?"
607 MS$(7)="DARAN HAENGT IMMER NOCH DAS
SEIL"
608 MS$(8)="MIT DEM SIE GEZOGEN WURDE."
609 MS$(9)="WENN ERST MAL DIE GIER ERWAC
HT, DER TOD AUCH OFT GESCHAEFTE MACHT."
698 :
699 REM ------------ 2, TITEL: EINLEITUNG
700 PRINT"□":POKE53280,0:POKE53281,0:PRI
NT"■";CHR$(14);
710 PRINT"│ERZLICH ○ILLKOMMEN ZUR ╲INIVE
RSION VON"
715 PRINTSPC(14)"■▥│┌└‾_♣,♥‾│■"
720 PRINT"■────────────────────────────────
────────"
725 PRINT"■♣UF DER ♥UCHE NACH DEM │LUECK
 IN DER"
730 PRINT"NEUEN ○ELT BEGEGNETEN ♥IE VOR
EINIGEN"
735 PRINT"│AGEN EINEM ALTEN TODKRANKEN ╲
ANN, DEM"
740 PRINT"♥IE IN SEINEN LETZTEN ♥TUNDEN
│EISTAND"
745 PRINT"LEISTETEN."
750 PRINT"♣US ‾ANKBARKEIT BERICHTETE ER
╮HNEN VON SEINER │OLDMINE UND DEN ";
755 PRINT"DORT VERSTECKTEN";:PRINT"_ESTE
N SEINES ╳ERMOEGENS."
760 PRINT"■♦AHLLOSEN │EFAHREN WIDERSTAND
EN ♥IE"
765 PRINT"AUF DEM ○EG DORTHIN; BALD WERD
EN ♥IE ╮HR";
770 PRINT"♦IEL ERREICHT HABEN UND ES WIR
D SICH"
775 PRINT"ZEIGEN, OB DER ♣LTE DIE ○AHRHE
IT GE-"
780 PRINT"SPROCHEN ODER IM _IEBERTRAUM G
EREDET"
```

```
785 PRINT"HATTE."
790 PRINT"————————————————————————————————
————————"
799 :
800 REM ------------- SPIELDATEN EINLESEN
810 FOR I=1 TO AV
815 READ VERB$(I):VERB$(I)=LEFT$(VERBS$(
I),WL)
820 NEXT I
830 FOR OBJEKT=1 TO AO
835 READ OB$(OBJEKT), RN$(OBJEKT), OB(OB
JEKT):RN$(OB)=LEFT$(RN$(OB),WL)
840 NEXT OBJEKT
845 FOR RAUM=1 TO AR
850 READ RAUM$(RAUM)
855 FOR RICHTUNG=1 TO 6
860 READ DURCHGANG(RAUM,RICHTUNG)
865 NEXT RICHTUNG
870 NEXT RAUM
875 :
880 PRINT"▣      OUENSCHEN ♠IE ─ATSCHLAEG
E FUER"
885 INPUT"   ⌐HR WEITERES ×ORGEHEN
 ";EI$
890 IF EI$="J" THEN GOSUB 900
895 GOTO 1000
899 :
900 REM ---------- 3. TITEL: INSTRUKTIONEN
910 PRINT"□";CHR$(14);
920 PRINT"  ─64 - ♣DVENTURE ♠YSTEM, ×ERS
ION 1.0"
930 PRINT"      (C) 1984  BY OALKOWIAK
"
940 PRINT"————————————————————————————————
————————"
950 PRINT"♠TELLEN ♠IE SICH EINEN ─OBOTER
 VOR, DEN"
951 PRINT"♠IE MIT ZAHLREICHEN ╯OMMANDOS
STEUERN"
952 PRINT"KOENNEN. ⌐CH BIN DIESER ─OBOTE
R, UND"
```

```
953 PRINT"ICH WERDE MICH FUER ♠IE DEN |E
FAHREN"
954 PRINT"DER VERWEGENSTEN ♣BENTEUER AUS
SETZEN."
955 PRINT"▓⌐AMIT ♠IE MICH SINNVOLL AGIER
EN LASSEN"
956 PRINT"KOENNEN, WERDE ICH ⌐HNEN DIE ♠
ITUATION,";
957 PRINT"IN DER ICH MICH GERADE BEFINDE
, JEWEILS GENAU BESCHREIBEN.";
958 PRINT" ♣NSCHLIESSEND SAGEN":PRINT"♠I
E MIR MIT ZWEI ○ORTEN, WIE ZUM |EI-"
959 PRINT"SPIEL ╱╱|‾_♠╱-|‾ | ╱‾_, ╱╲╲╲ ╲‾
, WAS"
960 PRINT"ICH TUN SOLL."
961 PRINT"▓⌐ARUEBER HINAUS VERSTEHE ICH
DIE |E-"
962 PRINT"FEHLE ⌐╱╳ENTUR   ♠♣╳‾    ⊔‾♣‾
"
963 PRINT"   ╳⊓╱ABELN    |⊓⊔⊓   ‾╱‾‾
"
964 PRINT"      ♠⊤⊤RE UND   ⌐╱♠TRUKTIONE
N"
975 PRINT"──────────────────────────────
─────────"
980 PRINT"▓▓"SPC(11)"(|ASTE DRUECKEN)";
985 GET EI$: IF EI$="" THEN 985
990 PRINT"▯":PRINTCHR$(142):RETURN
995 :
999 REM -------- BEGINN ADVENTURE-DRIVER
1000 PRINT"▯":PRINTCHR$(142)
1010 LEERZEILE$="
                                        "
1020 DATA NORDEN, SUEDEN, WESTEN, OSTEN,
 OBEN, UNTEN
1030 FOR RICHTUNG=1 TO 6
1040 READ RICHTUNG$(RICHTUNG)
1050 NEXT RICHTUNG
1070 PRINT"▯":POKE 53280,0:POKE 53281,0:
PRINT"▪"
1080 PRINT"▯" : ZUG=ZUG+1 : LW=LW+1
```

```
               : REM BEGINN NEUER ZUG --
1084 IF LW=LM THEN GOSUB 3000
1085 IF WERTUNG=WMAX THEN GOTO 4800
1090 POKE211,0:POKE214,0:SYS 58732
1100 FOR ZEILE=1 TO 10
1110 PRINT LEERZEILE$
1120 NEXT ZEILE
1130 POKE211,0:POKE214,0:SYS 58732
1131 REM ------------------- KEIN LICH
T
1132 IF LICHT =-1 THEN 1140
1133 PRINT"ICH WEISS NICHT GENAU, WO ICH
 BIN."
1134 PRINT"ES IST ZU DUNKEL, UM ETWAS ZU
 SEHEN."
1135 PRINT"▮AUCH DIE AUSGAENGE SEHE ICH
NICHT MEHR.":GOTO1330
1140 PRINT"ICH BIN ";
1150 PRINTRAUM$(SPIELER)
1160 PRINT"ICH SEHE ";:GEDRUCKT=0
1170 FOR I=1 TO AO
1180 IF OB(I)<>SPIELER THEN 1210
1190 IF POS(0)+LEN(OB$(I))+2<39 THEN PRI
NT OB$(I);", ";:GEDRUCKT=-1:GOTO 1210
1200 IF POS(0)+LEN(OB$(I))+2>=39 THEN PR
INT : GOTO 1190
1210 NEXT I:IF NOT GEDRUCKT THEN PRINT"N
ICHTS BESONDERES ";
1220 PRINT"▮■▮."
1230 PRINT LEERZEILE$
1240 PRINT";JICH KANN NACH ";
1250 FOR RICHTUNG=1 TO 6
1260 IF DURCHGANG(SPI,RICHTUNG)=0 THEN G
OTO 1310
1270 IF POS(0)=14 THEN PRINT RICHTUNG$(R
ICHTUNG);:GOTO 1310
1280 IFPOS(0)+LEN(RI$(RICHTUNG))<37 THEN
PRINT", ";RI$(RICHTUNG);:GOTO 1310
1290 IFPOS(0)+LEN(RI$(RICHTUNG))>=37 THE
N PRINT",":PRINT RI$(RICHTUNG);:GOTO1310
1300 IF POS(0)<16 AND POS(0)>2 THEN PRIN
```

```
T", ";RICHTUNG$(RICHTUNG);:GOTO 1310
1310 NEXT RICHTUNG
1320 PRINT"."
1330 PRINT"■______________________________________
_______"
1340 IF WERTUNG=IMAX THEN GOTO 4800
1350 IF SP=15 THEN MS$(0)="LEIDER WAR DO
RT DER BAER.":FORI=1TO1000:NEXT:GOTO4500
1360 IF LW<=0 THEN LI=0
1370 IF EX=ZUG AND SP=20THENMS$(0)="RUMM
S ! - AUCH MICH HATS ZERRISSEN.":GOTO450
0
1380 IF SP>18 AND OB(23)<>-1 THEN LICHT=
0
1390 POKE 211,0:POKE 214,24:SYS 58732:PR
INT"■";:INPUT"WAS SOLL ICH TUN";EI$:PRIN
T"■■";
1395 RL=LM-LW:IF (RL>0ANDRL<15)THEN PRIN
T "IN";LM-LW;"ZUEGEN STEHE ICH IM DUNKLE
N."
1400 IF LEN(EI$)>2 THEN 1500
1410 IFEI$="N"ANDDURCHGANG(SPI,1)<>0THEN
SPI=DURCHGANG(SPI,1):PRINT"O.K.":GOTO108
0
1420 IFEI$="S"ANDDURCHGANG(SPI,2)<>0THEN
SPI=DURCHGANG(SPI,2):PRINT"O.K.":GOTO108
0
1430 IFEI$="W"ANDDURCHGANG(SPI,3)<>0THEN
SPI=DURCHGANG(SPI,3):PRINT"O.K.":GOTO108
0
1440 IFEI$="O"ANDDURCHGANG(SPI,4)<>0THEN
SPI=DURCHGANG(SPI,4):PRINT"O.K.":GOTO108
0
1450 IFEI$="OB"ANDDURCH(SPI,5)<>0THENSPI
=DURCHGANG(SPI,5):PRINT"O.K.":GOTO1080
1460 IFEI$="U"ANDDURCHGANG(SPI,6)<>0THEN
SPI=DURCHGANG(SPI,6):PRINT"O.K.":GOTO108
0
1470 PRINT"DAHIN FUEHRT KEIN WEG !":GOTO
 1080
1490 IF LEN(EINGABE$)>6 THEN GOTO 2000
```

```
1498 :
1499 REM ------- START INVENTUR --------
1500 IF LEFT$(EINGABE$,3)<>"INV" THEN GO
TO 1560
1510 PRINT"ICH TRAGE FOLGENDES MIT MIR:"
1520 FOR I=1 TO AO
1530 IF OB(I)=-1 THEN PRINT OB$(I)
1540 NEXT I
1550 GOTO 1080
1551 REM -------- ENDE INVENTUR --------
1559 REM ------------------------- SCORE
 LIST170
1560 IF LEFT$(EINGABE$,3)<>"SCO" THEN GO
TO 1600
1561 PRINT"VON";WMAX;"PUNKTEN HAST DU IN
";ZUG;"ZUEGEN"
1562 PRINT WERTUNG;"PUNKTE ERREICHT ! D
AS ENTSPRICHT"
1563 PRINT"EINEM SCHNITT VON";WERT/ZUG;"
PUNKTEN."
1565 GOTO1080
1599 REM -------------------- SAVE GAME
1600 IF LEFT$(EINGABE$,4)<>"SAVE" THEN G
OTO 1700
1605 PRINT"⌐"SPC(10)"SPIELSTAND SPEICHER
N":INPUT"■■UNTER WELCHEM NAMEN SPEICHERN
";EI$
1610 IF LEN(EI$)>16 THEN 1605
1615 PRINT"■■SPEICHERUNG VON ";EI$;" LAE
UFT ";
1620 OPEN 2,8,2,"§0:"+EI$+",S,W"
1625 PRINT#2,SPIELER
1627 PRINT#2, WERTUNG:PRINT#2,L1:PRINT#2
,LM:PRINT#2,LW
1628 PRINT#2, ZUG
1630 FOR I=1 TO AO
1631 PRINT#2,OB(I)
1632 NEXT I
1633 PRINT".";
1635 FOR RAUM=1 TO AR
1636 FOR RICHTUNG=1 TO 6
```

```
1637 PRINT#2, DURCHGANG(RA,RI)
1638 NEXT RICHTUNG
1639 NEXT RAUM
1640 PRINT".";
1645 FOR I=1 TO AF
1646 PRINT#2,FL(I)
1647 NEXT I
1648 PRINT".";
1650 CLOSE 2
1660 GOSUB 1680
1670 PRINT"□":GOTO 1080
1678 :
1679 REM --------------- DISKETTENFEHLER
1680 OPEN1,8,15
1681 INPUT#1,A,B$,C,D
1682 IFA<>0THENPRINT:PRINT"■■■■ACHTUNG:
":PRINTB$:FORI=1TO 5000:NEXT:CLOSE2:CLOS
E1:GOTO1080
1683 CLOSE1
1684 RETURN : REM ---------- ENDE FEHLER
1698 :
1699 REM ----------------------- LOAD GAME
1700 IF LEFT$(EINGABE$,4)<>"LOAD" THEN G
OTO 1800
1705 PRINT"□"SPC(10)"ALTES SPIEL LADEN":
INPUT"■■WELCHEN SPIELSTAND LADEN";EI$
1710 IF LEN(EI$)>16 THEN 1805
1715 PRINT"■■"EI$" WIRD GELADEN.";
1720 OPEN 2,8,2,EI$+",S,R"
1725 INPUT#2,SPIELER
1726 PRINT".";
1727 INPUT#2, WERTUNG:INPUT#2,L1:INPUT#2
,LM:INPUT#2,LW
1728 INPUT#2, ZUG
1730 FOR I=1 TO AO
1731 INPUT#2,OB(I)
1732 NEXT I
1733 PRINT".";
1735 FOR RAUM=1 TO AR
1736 FOR RICHTUNG=1 TO 6
1737 INPUT#2, DURCHGANG(RA,RI)
```

```
1738 NEXT RICHTUNG
1739 NEXT RAUM
1740 PRINT".";
1745 FOR I=1 TO AF
1746 INPUT#2,FL(I)
1747 NEXT I
1748 PRINT".";
1750 CLOSE 2
1760 GOSUB 1680
1770 PRINT"□":GOTO 1080
1771 REM -------------- ENDE LOAD GAME
1778 :
1798 :
1799 REM -------------------- VOKABULAR
1800 IF LEFT$(EINGABE$,3)<>"VOK" THEN GO
TO 1900
1805 PRINT"□":PRINT"■ICH VERSTEHE FOLGEN
DE VERBEN■■"
1806 RESTORE
1810 FOR I=1 TO AV
1820 READ VO$:PRINT VO$
1830 NEXT I
1840 GOSUB 1890
1845 PRINT"□":PRINT"UND FOLGENDE OBJEKTE
 SIND MIR BEKANNT:■"
1849 ZEILE=0
1850 FOR I=1 TO AO
1855 ZEILE=ZEILE+1
1860 READ VO$,VO$,X:PRINTVO$
1865 IF ZEILE=20 THEN GOSUB 1890
1866 IF ZEILE=20 THEN ZEILE=1 : PRINT"□"
1870 NEXT I
1880 GOSUB 1890:PRINT"□":GOTO 1080
1890 PRINTSPC(24)"■TASTE DRUECKEN";
1895 GETEI$:IFEI$=""THEN 1895
1896 PRINT"□":RETURN
1900 IF LEFT$(EINGABE$,3)<>"INS" THEN GO
TO 1950
1910 GOSUB 900
1920 GOTO 1080
1950 IF LEFT$(EINGABE$,3)<>"END" THEN GO
```

```
TO 1960
1955 PRINT"□DER AUTOR WUENSCHT IHNEN FUE
RS NAECHSTE■MAL MEHR ERFOLG !■■■■■":END
1959 REM ---------------------------- HELP
READY.
1960 IF LEFT$(EINGABE$,4)<>"HELP" THEN G
OTO 2000
1970 IF SP=4 AND OB(10)=0 THEN PRINT "FA
ST WAERE ICH IN EINE GRUBE GEFALLEN.":GO
TO 1080
1971 IF SP=4 AND OB(11)=SP AND NOTFL(2)
THEN PRINTMS$(6):GOTO1080
1975 PRINT"ERST SEHEN, DANN DENKEN UND Z
ULETZT":PRINT"HANDELN !":GOTO 1080
1979 REM ---------------------- ENDE HELP
1990 :
1999 REM ------------ ANALYSE DER EINGABE
2000 LN=LEN(EI$)
2010 FOR EL=1 TO LN
2020 TEST$=MID$(EI$,EL,1)
2030 IF TEST$<>" "THEN NEXT EL
2040 EV$=LEFT$(EI$,WL)
2050 RL=LN-EL
2060 IF RL<0 THEN 2090
2070 EO$=RIGHT$(EI$,RL)
2080 EO$=LEFT$(EO$,WL)
2090 FOR VN=1 TO AV
2100 IF EV$=VERB$(VN) THEN 2130
2110 NEXT VN
2120 PRINT"DAS VERB VERSTEHE ICH NICHT!"
:GOTO 1080
2130 FOR N=1 TO AO
2140 IF EO$=RN$(N) THEN 2200
2150 NEXT N
2160 PRINT"ICH VERSTEHE DAS OBJEKT NICHT
!":GOTO 1080
2169 :
2170 REM -- SPRUNGZIELE FUER AUSFUEHRUNG
2180 REM  UNTERSUCHE, NIMM, LEG, OEFFNE,
 BENUTZE, ZERSTOERE, ZUENDE, FUELLE
2200 ON VN GOTO 5000,2210,7000,8000,9000
```

```
,10000,11000,12000,13000,14000,15000
2201 :
2210 ANZAHL=0
2220 FOR I=1 TO AO
2230 IF OB(I)=-1 THEN ANZAHL=ANZAHL+1
2240 IF ANZAHL=IMAX THEN PRINT"NEIN DANK
E. - ICH TRAGE SCHON GENUG !":GOTO1080
2250 NEXT I
2260 GOTO 6000
2270 REM -------- ENDE PROBE PLATZ IN INV
2999 REM ------------UP LICHTSCHALTER
3000 IF LICHT=-1 THEN LICHT=0
3010 IF LICHT=0 THEN LICHT=-1
3020 LW=0
3030 RETURN
3040 REM ----------- ENDE LICHTSCHALTER
4498 :
4499 REM -------------------- SPIELENDE
4500 PRINT"":REM ------ SPIELER TOD ----
4600 PRINT "AUCH DAS NOCH !":PRINT:PRINT
MS$(0)
4610 PRINT"ICH BIN TOD !":PRINT
4620 INPUT"SOLL ICH ES NOCH EINMAL VERSU
CHEN";EI$
4630 IF LEFT$(EI$,1)="J"THEN RUN
4640 PRINT"":END
4641 :
4799 :
4800 PRINT"":REM ------ SPIELER SIEGT -
4805 EW=WERTUNG/ZUG
4810 PRINT"HERZLICHEN GLUECKWUNSCH !"
4815 PRINT"SIE HABEN DIE IHNEN GESTELLTE
 AUFGABE"
4820 PRINT"GELOEST."
4825 PRINT"IN";ZUG;"SPIELZUEGEN HABEN SI
E PRO ZUG"
4830 PRINT EW;"PUNKTE GEMACHT."
4835 PRINT"DAMIT HABEN SIE EIN ";
4840 IF EW<.5 THEN MS$(0)="MISERABLES"
4845 IF EW>.5 THEN MS$(0)="MAESSIGES"
4850 IF EW>1.0 THEN MS$(0)="GUTES"
```

```
4855 IF EW>1.5 THEN MS$(0)="SEHR GUTES"
4860 PRINTMS$(0);" ERGEBNIS"
4865 PRINT"ERZIELT."
4899 END
4998 :
4999 REM --------- SPIELERZUG AUSFUEHREN
5000 IF OB(N)<>SP AND OB(N)<>-1 THEN GOT
O 5900          REM  ** UNTERSUCHE **
5002 IF N=1 THEN PRINTMS$(1):GOTO1080
5003 IF N=3 THEN PRINTMS$(1):GOTO1080
5004 IF N=4 THEN PRINT"IN EINER ECKE STE
HT EIN REGAL.":OB(8)=SP:GOTO1080
5005 IF N=5 AND NOTFL(3) THEN PRINT "DIE
 FLASCHE IST GEFUELLT MIT HONIG.":GOTO10
80
5006 IF N=6 THEN PRINTMS$(1):GOTO1080
5007 IF N=7 ANDOB(9)=0THENPRINT"IN DER K
ISTE LIEGT SPRENGSTOFF.":GOTO1080
5008 IF N=8 AND OB(5)=0THEN PRINT"AUF DE
M REGAL STEHT EINE KORBFLASCHE.":OB(5)=S
P:GOTO1080
5009 IF N=8 AND OB(5)<>0 THEN PRINT MS$(
1):GOTO1080
5010 IF N=10THEN PRINT"IN DEM ERDLOCH LI
EGT EINE EISENTRUHE.":OB(11)=SP:GOTO1080
5011 IFN=11ANDNOTFL(1) THENPRINT"SIE IST
 MIT EINER EISENKETTE VERSCHLOSSEN.":GOT
O1080
5012 IF N=11ANDFL(1)ANDNOTFL(2)THENPRINT
"VON AUSSEN SEHE ICH NICHTS BESONDERES."
:GOTO1080
5013 IF N=11ANDFL(1)ANDFL(2)THENPRINT"SI
E IST VOLLER SILBERMUENZEN.":OB(12)=SP:G
OTO1080
5014 IF N=12 THEN PRINT"GENAU DAS SUCHE
ICH !!":GOTO1080
5015 IFN=13THENPRINT"ICH HABE EINEN BAER
EN AUFGESCHRECKT.":OB(14)=SP:GOTO1080
5017 IF N=15 THENPRINT"ZWISCHEN DEN BUES
CHEN IST EIN ERDLOCH.":OB(10)=SP:GOTO108
0
```

```
6014 IF N=12 AND OB(N)=4 THEN PRINT "O.K
.":OB(12)=-2:RN$(12)="*":WE=WE+10:GOTO10
80
6015 IF N=1 AND SP=5 THEN MS$(0)="DER BA
ER HAT MICH ERSCHLAGEN.":GOTO4500
6016 IF N=16 THEN PRINT"O.K.":OB(18)=-1:
GOTO1080
6017 IFN=17ANDOB(N)=5ANDFL(3)THENPRINT"O
.K.":OB(17)=-2:RN$(N)="*":WE=WE+10:GOTO1
080
6018 IF N=17 AND NOT FL(3) THEN MS$(0)="
DER BAER STUERZT SICH AUF MICH.":GOTO450
0
6019 IF N=12 AND OB(N)=-1 THEN PRINT"DAS
 SILBER HABE ICH BEREITS !":GOTO 1080
6020 IF N=17 AND OB(N)=-1 THEN PRINT"ICH
 BIN BEREITS IM BESITZ DES GOLDES !":GOT
O 1080
6021 IF N=19 THEN PRINTMS$(2):GOTO1080
6022 IF N=21 AND FL(4) THEN PRINT"O.K.":
OB(N)=-1:GOTO1080
6023 IF N=22 THEN PRINT"O.K.":OB(N)=-1:G
OTO1080
6024 IF N=24 THEN PRINT"O.K.":OB(N)=-1:G
OTO1080
6025 IF N=25 THEN PRINT"ER STECKT ZU TIE
F IM FELS.":GOTO1080
6026 IF N=27 THEN PRINTMS$(3):GOTO1080
6027 IF N=28 THEN PRINT"UND WAS SOLL ICH
 DAMIT ?":GOTO1080
6030 IFN=31ANDOB(5)<>-1THENPRINT"WOMIT D
ENN ?":GOTO1080
6031 IFN=31ANDOB(5)=-1THENPRINT"O.K. - D
IE FLASCHE IST VOLL.":FL(6)=-1:GOTO1080
6032 IF N=32THENPRINT"O.K. - OH, WAS IST
 DAS DENN ?":FL(5)=-1:GOTO1080
6033 IF N=33 AND OB(N)=SP THEN PRINT"O.K
.":OB(N)=-2:RN$(N)="*":WE=WE+10:GOTO1080
6034 IF N=34 AND OB(N)=SP THEN PRINT"O.K
.":OB(N)=-2:WE=WE+10:RN$(N)="*":GOTO1080
6035 IFN=35THENPRINT"ES SIND GAR KEINE S
```

```
5037 IFN=32ANDFL(4)ANDOB(33)<>-2THENPRIN
T"SIE LAG AUF SILBERSTUECKEN.":OB(33)=SP
:GOTO1080
5899 PRINTMS$(1):GOTO1080
5900 REM  GEGENSTAND NICHT VORHANDEN
5901 IF N=1 AND SP=2 THEN PRINTMS$(1):GO
TO1080
5902 IFN=6 AND OB(5)=-1 THEN PRINT"ER IS
T SUESS UND GUT.":GOTO1080
5903 IF N=6 AND OB(5)<>-1 THEN PRINT"ICH
 HABE KEINEN HONIG !":GOTO1080
5904 IFN=9AND(OB(7)=SPOR OB(7)=-1)THENPR
INT"ER SIEHT SEHR EXPLOSIV AUS !":GOTO10
80
5905 IF M=9 THEN PRINT"DER SPRENGSTOFF S
IEHT BEDROHLICH AUS !":GOTO1080
5910 IFN=1ANDSP=5ANDOB(14)=5THENPRINT"IC
H SCHEINE SEINEN APPETIT ANZUREGEN !":GO
TO1080
5911 IF N=20 AND SP=22 THENPRINTMS$(9):G
OTO1080
5912 IF N=44 AND SP=23 THEN MS$(0)="ICH
BIN IM TOTENREICH.":GOTO4500
5990 PRINT "SO ETWAS SEHE ICH HIER NICHT
 !" : GOTO 1080
6000 IF OB(N)<>SP AND OB(N)<>-1 THEN GOT
O 6900
6001 IF N=1 THEN PRINT MS$(2):GOTO 1080
6002 IF N=3 THEN PRINT MS$(2):GOTO 1080
6003 IF N=4 THEN PRINT MS$(3):GOTO 1080
6004 IF N=8 THEN PRINT MS$(2):GOTO 1080
6005 IF N=11THEN PRINT MS$(2):GOTO 1080
6006 IF N=10THEN PRINT MS$(3):GOTO 1080
6007 IF N=13THEN PRINT MS$(3):GOTO 1080
6008 IF N=15THEN PRINT MS$(2):GOTO 1080
6010 IF N=5 THEN OB(5)=-1:PRINT"O.K.":GO
TO1080
6011 IF N=6 THEN OB(5)=-1:PRINT"O.K.":GO
TO1080
6012 IF N=7 THEN OB(7)=-1:PRINT"O.K.":GO
TO1080
```

```
5018 IF N=16 THEN PRINT"SIE SEHEN SEHR S
TABIL AUS.":GOTO1080
5019 IF N=17 THEN PRINT"ES HANDELT SICH
UM REINES GOLD !":GOTO1080
5020 IF N=19 AND OB(21)=0 THEN PRINTMS$(
7):PRINTMS$(8):OB(21)=SP:GOTO 1080
5021 IF N=23 THEN PRINT"ES IST EINE ALTE
 OELLAMPE.":GOTO1080
5022 IF N=24 THEN PRINT"IN DER WAND HAEN
GT STECKT EIN HAKEN.":OB(25)=SP:GOTO1080
5023 IF N=25 THEN PRINT"TUT MIR LEID, ER
 STECKT ZU FEST."GOTO1080
5024 IF N=26 THENMS$(0)="ER IST SEHR TIE
F - UND ICH WAR ZU DICHT AM RAND.":GOTO4
500
5025 IF N=27 THEN PRINT"IHR TISCHLER WAR
 EIN KOENNER.":GOTO1080
5026 IF N=28 AND OB(N)=0THEN PRINT"ICH H
ABE ZWEI SAECKE ENTDECKT.":OB(29)=SP:GOT
O1080
5027 IF N=29 THEN PRINT"SIE SIND GEFUELL
T MIT GOLDSTAUB.":GOTO1080
5028 IF N=31 THEN PRINT"SIE IST OELIG SC
HMIERIG UND KLEBT AN DENFINGERN.":GOTO10
80
5029 IFN=32THENPRINT"SIE STINKT !"
5030 IFN=32ANDOB(34)=0THEN PRINT"IN DER
JACKENTASCHE SIND GOLDMUENZEN.":OB(34)=S
P:GOTO1080
5031 IF N=35 THEN PRINT"ES SIND GAR KEIN
E SPINNWEBEN.":OB(N)=0:OB(42)=SP
5032 IF N=35 THEN PRINT"ES IST EINE ZUEN
DSCHNUR.":GOTO1080
5033 IF N=36 THEN PRINT"ES IST TATSAECHL
ICH REINES GOLD.":GOTO1080
5034 IF N=37 THEN PRINT"DARAUF STEHT:":P
RINTMS$(9):GOTO1080
5035 IF N=42 THEN PRINT"SIE FUEHRT HINAU
F ZUR DECKE.":GOTO1080
5036 IF N=43 THEN PRINT"ES IST EINS DER
UEBLICHEN WESTERFEUER   ZEUGE."
```

```
PINNWEBEN, ES WAR"
6036 IFN=35THENPRINT"EINE ZUENDSCHNUR.":
OB(N)=0:OB(42)=-1:GOTO1080
6037 IF N=37 THEN OB(N)=-1:PRINT"O.K.":G
OTO1080
6038 IF N=41 THEN OB(N)=-1:PRINT"O.K.":G
OTO1080
6039 IF N=42 THEN OB(N)=-1:PRINT"O.K.":G
OTO1080
6040 IF N=43 THEN OB(N)=-1:PRINT"O.K.":G
OTO1080
6041 IF N=21 AND NOT FL(4) THEN PRINT"ES
 IST AN DER LORE BEFESTIGT.":GOTO1080
6042 IF N=23 THEN PRINT"O.K.":OB(N)=-1:G
OTO1080
6043 IF N=29 AND OB(N)=SP THEN PRINT"O.K
.":OB(N)=-2:RN$(N)="*":WE=WE+10:GOTO1080
6044 IF N=38 AND OB(N)=SP THEN PRINT"O.K
.":OB(N)=-2:RN$(N)="*":WE=WE+10:GOTO1080
6900 IF N=9THENMS$(0)="BEI DER BERUEHRUN
G IST DER SPRENGSTOFF  EXPLODIERT.":GOTO
4500
6910 IF N=20 AND SP=22 THENPRINTMS$(9):G
OTO1080
6998 PRINT "ICH VERSTEHE NICHT, WAS DU M
EINST !":GOTO1080
7000 IF N=16 AND OB(18)=-1 THEN PRINT"O.
K.":OB(18)=SP:GOTO1080
7005 IF OB(N)<>-1 THEN PRINT"SO ETWAS HA
BE ICH DOCH GAR NICHT.":GOTO1080
7010 IF N=6ANDSP=5THENOB(6)=0:FL(3)=-1:P
RINTMS$(4):PRINTMS$(5):OB(14)=15:OB(5)=1
4:GOTO1080
7020 IFN=5ANDSP=5 THENOB(5)=14:FL(3)=-1:
PRINTMS$(4):PRINTMS$(5):OB(14)=15:GOTO10
80
7030 IF N=17 THEN OB(N)=SP:WE=WE-10:PRIN
T"O.K.":GOTO1080
7900 OB(N)=SP:PRINT"O.K.":GOTO1080
8000 IF OB(N)<>SP AND OB(N)<>-1 THEN PRI
NT"SO ETWAS IST HIER NICHT.":GOTO1080
```

```
8005 IF N=4 AND SP=3 THEN PRINT"DIE HUET
TE WAR BEREITS OFFEN.":GOTO1080
8010 IF N=5 THEN PRINT"O.K.":GOTO1080
8020 IF N=11 AND NOT FL(1) THEN PRINT"DA
S LAESSST DIE KETTE NICHT ZU.":GOTO1080
8025 IF N=11 AND FL(1) THEN PRINT"O.K. -
 DER DECKEL KLAPPT NACH HINTEN.":FL(2)=-
1:GOTO1080
8030 IF N=23 THEN PRINT"O.K.":GOTO1080
8035 IF N=29 THEN PRINT"O.K.":GOTO1080
8040 IF N=32 THEN PRINT"TUT MIR LEID !":
PRINT"ICH BIN NICHT FRANKENSTEIN.":GOTO1
080
8045 IF N=36 THEN PRINT"WIE ?":GOTO1080
8999 PRINT"ICH VERSTEHE NICHT, WAS DU ME
INST.":GOTO1080
9000 IF OB(N)<>SP AND OB(N)<>-1 THEN GOT
O 9900            REM  ** UNTERSUCHE **
9020 IF N=19 THEN PRINT"WIE UND WOZU ?":
GOTO1080
9030 IF N=21 THEN PRINT"WIE UND WOZU ?":
GOTO1080
9900 IF N=16 AND OB(18)=-1AND SP=4 THEN
PRINT "DIE KETTE ZERSPRINGT.":FL(1)=-1:G
OTO 1080
9999 PRINT"ICH VERSTEHE NICHT, WAS DU ME
INST.":GOTO1080
10000 IFN=16ANDSP=4 AND OB(18)=-1 THEN P
RINT "DIE KETTE ZERSPRINGT.":FL(1)=-1: G
OTO 1080
10010 IF N=19 AND SP=4 AND OB(18)<>-1 TH
EN PRINT"WOMIT ?":FL(1)=-1:GOTO1080
10020 IF N=36 AND OB(22)=-1 THEN PRINT"O
.K.":DU(22,1)=23:GOTO1080
10030 IF N=21 AND(OB(N)=SP OR OB(N)=-1)T
HENPRINT"O.K.":FL(4)=-1:GOTO1080
10040 IF N=27 AND OB(22)=-1 THEN PRINT"O
.K.":DU(10,2)=12:GOTO1080
10050 IF N=27 AND OB(22)<>-1 THEN PRINT"
WOMIT ?":GOTO1080
10999 PRINT"ICH VERSTEHE NICHT, WAS DU M
```

```
EINST.":GOTO1080
11000 IF OB(43)<>-1 THEN PRINT"ICH HABE
NICHTS ZUM ZUENDEN.":GOTO1080
11010 IF N=8 AND LZ<=0 THEN PRINT"IN DER
 LAMPE IST KEIN OEL MEHR !":GOTO1080
11020 IFN=42AND(OB(43)=-1OR LIGHT)THENPR
INT"ZZZIISCHH":FL(7)=-1:EX=ZUG+3:OB(N)=0
11021 IF N=42AND(OB(43)=-1 OR LI)THENDU(
30,2)=31:DU(31,1)=30:GOTO1080
11030 IF N=23AND OB(23)=-1 THEN PRINT"O.
K. - DIE LAMPE BRENNT.":LM=L1:LW=1:GOTO1
080
11040 IFN=43 THEN PRINT"O.K. - DER DOCHT
 GLIMMT.":GOTO1080
11998 PRINT"ICH VERSTEHE NICHT, WAS DU M
EINST.":GOTO1080
12000 REM ---------------- AKTION FUELLE
12010 IF N=23AND(FL(6)ANDOB(23)=-1)THENL
M=50:FL(6)=0:LW=0: PRINT"O.K.":GOTO1080
12020 IF N=5 AND SP=17 THEN FL(6)=-1:PRI
NT"DIE FLASCHE IST VOLL.":GOTO1080
12030 IF N=43 AND OB(43)=-1 THEN PRINT"D
ER DOCHT GLIMMT.":GOTO1080
12998 PRINT"ICH VERSTEHE NICHT, WAS DU M
EINST.":GOTO1080
13000 REM ---------------- AKTION BETRETE
13010 IF N=13 AND SP=5 THEN SP=12: PRINT
 "O.K.":GOTO 1080
13998 PRINT"ICH VERSTEHE NICHT, WAS DU M
EINST."GOTO1080
14000 IF N=23 AND OB(N)=-1 THEN L1=LM-LW
:PRINT"O.K.":GOTO1080
14998 PRINT"ICH VERSTEHE NICHT, WAS DU M
EINST."GOTO1080
15000 IFN=21ANDSP=9THENPRINT"O.K.":OB(N)
=9:DU(9,6)=18:DU(18,5)=9:GOTO1080
15998 PRINT"ICH VERSTEHE NICHT, WAS DU M
EINST.":GOTO1080

READY.
```

```
3 REM  DAS VERZAUBERTE SCHLOSS
4 REM  (C) 1984 BY J.WALKOWIAK
8 REM"────────────────────────────────────
10 PRINT"▯":POKE 53280,0:POKE 53281,0:PR
INT"▪":AR=26:AO=45:AV=8:SP=9
20 PRINT"▪ WILLKOMMEN IM ▪VERZAUBERTEN
SCHLOSS . "
21 PRINT"▪ VERSION 1.0    (C) 1984 BY W
ALKOWIAK ▪"
25 PRINT"───────────────────────────────
───────"
30 PRINT:PRINT: PRINT"ICH BIN DEIN DIR T
REU ERGEBENER DIENER.":PRINT
35 PRINT"SAGE MIR MIT KURZEN ANWEISUNGEN
 WIE:":PRINT
40 PRINT"▪UNTERSUCHE TISCH▪, ▪LEG FLASCH
E WEG▪":PRINT
50 PRINT"USW. WAS ICH TUN SOLL.":PRINT
60 PRINT"WENN DU WISSEN WILLST, WAS ICH
BEI MIR":PRINT
65 PRINT"HABE, LASS MICH ▪INVENTUR▪ MACH
EN."
66 PRINT"───────────────────────────────
───────"
70 PRINT:PRINT
80 PRINT"              BIST DU BEREIT ?"
85 GETA$:IFA$=""THEN85
90 DIM RAUM$(AR), DURCHGANG(AR,6), OB$(A
O), RN$(AO), OB(AO),MS$(11)
101 DATA"IN DER SCHLOSSBIBLIOTHEK.",0,3,
0,2,0,0
102 DATA"IM STUDIERZIMMER.",0,0,1,0,0,0
103 DATA"IN DER KUECHE.",1,8,0,4,0,0
104 DATA"IM SCHLOSSHOF.",0,8,3,6,0,0
```

```
105 DATA"IN EINEM DER WACHTUERME.",0,0,0
,0,0,11
106 DATA"IM SCHLOSSHOF.",11,0,4,0,0,0
107 DATA"VOR DER ZUGBRUECKE.",0,0,6,0,0,
0
108 DATA"IM GROSSEN FESTSAAL.",4,8,3,9,0
,0
109 DATA"IN EINEM PRIVATGEMACH.",9,9,8,0
,0,0
110 DATA"AUF EINER ZUGBRUECKE.",0,0,7,12
,0,0
111 DATA"IM SCHLOSSHOF.",0,6,0,0,5,0
112 DATA"AUF EINEM WALDWEG.",12,16,10,13
,0,0
113 DATA"IM ZAUBERWALD.",21,17,12,15,0,0
114 DATA"IN EINER DUESTEREN HOEHLE.",0,1
8,14,0,0,0
115 DATA"IM WALD.",15,15,15,16,0,0
116 DATA"IM ZAUBERWALD.",12,21,15,17,0,0
117 DATA"IM ZAUBERWALD.",13,22,16,18,0,0
118 DATA"AM FUSSE EINES GEBIRGES.",14,0,
17,19,0,0
119 DATA"IN EINER DUESTEREN HOEHLE.",25,
0,18,19,0,0
120 DATA"IN EINEM SCHLOSS.",20,20,20,21,
0,0
121 DATA"IM WALD.",16,21,20,21,0,0
122 DATA"IM SUMPF.",17,0,0,23,0,0
123 DATA"IM SUMPF.",0,0,22,24,0,0
124 DATA"AN EINER QUELLE.",23,24,0,23,0,
0
125 DATA"IN EINER DUESTEREN HOEHLE.",25,
19,25,14,14,0
126 DATA"IN DER SCHLANGENGRUBE.",0,0,0,0
,22,0
401 DATA"UNZAEHLIGE BUECHERREGALE","BUE"
,1
402 DATA"EINEN TISCH","TIS",1
403 DATA"MEINEN LEHRER DORE COMMO","DOR"
,2
404 DATA"EINEN GROSSEN TOPF","TOP",3
```

```
405 DATA"EINE FLASCHE","FLA",0
406 DATA"DEN SCHLOSSBRUNNEN","BRU",4
407 DATA"BUCH","BUC",0
408 DATA"SCHLAFENDE WAECHTER","WAE",5
409 DATA"DEN SEILZUG DER ZUGBRUECKE","SE
I",5
410 DATA"BEWOHNER DES SCHLOSSES","BEW",8
411 DATA"MEINE GAESTE","GAE",8
412 DATA"ZETTEL","ZET",0
413 DATA"EINEN KUECHENTISCH","KUE",3
414 DATA"EIN SCHLACHTERMESSER","MES",0
415 DATA"EINE ARMBRUST","ARM",5
416 DATA"EINEN SCHLUESSEL","SCH",0
417 DATA"EIN SCHWERES EISENTOR","TOR",6
418 DATA"EINE ZUGBRUECKE","ZUG",7
419 DATA"NICHTS BESONDERES","NIC",9
420 DATA"NICHTS BESONDERES","NIC",11
421 DATA"BAEUME","BAE",12
422 DATA"BAEUME","BAE",13
423 DATA"MEHRERE VOLLGEPACKTE REGALE","R
EG",14
424 DATA"NICHTS BESONDERES","NIC",25
425 DATA"BAEUME","BAE",15
426 DATA"BAEUME","BAE",16
427 DATA"BAEUME","BAE",17
428 DATA"EINE GROSSE ELSTER","ELS",17
429 DATA"EINEN HOEHLENEINGANG","HOE",18
430 DATA"GOLDEN GLAENZENDES METALL","MET
",19
431 DATA"EINE ALTE RUESTUNG","RUE",0
432 DATA"NICHTS BESONDERES","NIC",21
433 DATA"SUMPF","SUM",0
434 DATA"SCHLAMM","SCH",23
435 DATA"SCHLANGENEIER","EIE",26
436 DATA"DEN TRUEGERISCHEN BODEN","BOD",
23
437 DATA"SPRUDELNDES WASSER","WAS",24
438 DATA"EINEN SCHLOSSGRABEN","GRA",10
439 DATA"NICHTS BESONDERES","NIC",20
440 DATA"SPINNWEBEN","SPI",0
441 DATA"EINE FLASCHE MIT BLAUEM WASSER"
```

```
,"FLA",0
442 DATA"NICHTS BESONDERES.","NIC",19
443 DATA"EINE SCHLANGENGRUBE","GRU",22
444 DATA"SCHLANGEN","SCH",26
445 DATA"EIN STEINERNER ALTAR","ALT",25
800 DATA UNT,NIM,LIE,OEF,ZER,BEN,FUE,LEG
900 MS$(0)="ICH VERSTEHE NICHT, WAS DU M
EINST."
901 MS$(1)="ICH SEHE NICHTS BESONDERES."
902 MS$(2)="SO STARK BIN ICH NICHT."
903 MS$(3)="SIE SCHEINEN TIEF ZU SCHLAFE
N."
904 MS$(4)="DAS HALTE ICH NICHT FUER SIN
NVOLL."
905 MS$(5)="ICH SEHE, ZITTRIG GESCHRIEBE
N"
906 MS$(6)="DIE ELSTER HAT MIR ETWAS GES
TOHLEN !"
907 MS$(7)="DA LIEGT EINE ALTE RUESTUNG.
"
908 MS$(8)="IN DER HAND HAELT ER EINEN Z
ETTEL."
909 MS$(9)="IN DER TASCHE HAT ER EINEN S
CHLUESSEL."
910 MS$(10)="EINE INSCHRIFT BESAGT:"
911 MS$(11)="NIMM UNSER OPFER UND SEI UN
S GNAEDIG."
950 FOR RA=1 TO AR:READRA$(RA)
952 FOR RI=1 TO 6:READDU(RA,RI)
954 NEXT RI:NEXT RA
961 FOR OB=1 TO AO:READ OB$(OB), RN$(OB)
, OB(OB):NEXT OB
970 FOR I=1 TO AV:READ VE$(I):NEXT I
1010 LE$="
          "
1020 DATA NORDEN, SUEDEN, WESTEN, OSTEN,
 OBEN, UNTEN
1030 FOR RI=1 TO 6:READ RI$(RI):NEXT RI
1070 PRINT"□"
1080 PRINT"□":POKE211,0:POKE214,0:SYS587
32
```

```
1090 POKE211,0:POKE214,0:SYS 58732
1100 FOR Z=1 TO 10:PRINTLE$:NEXT Z
1130 POKE211,0:POKE214,0:SYS 58732
1140 PRINT"ICH BIN ";
1150 PRINTRAUM$(SP)
1160 PRINT"ICH SEHE ";
1170 FOR I=1 TO AO
1180 IF OB(I)<>SP THEN 1200
1185 IF POS(0)+LEN(OB$(I))+2<39 THEN PRI
NT OB$(I);", ";:GOTO 1200
1190 IF POS(0)+LEN(OB$(I))+2>39 THEN PRI
NT
1196 GOTO 1185
1200 NEXT I
1205 PRINT"▮▮▮."
1210 PRINT
1215 PRINT
1220 PRINT"▮ICH KANN NACH ";
1230 FOR RI=1 TO 6
1240 IF DU(SP,RI)=0 THEN GOTO 1250
1245 IFPOS(0)=14THENPRINTRI$(RI);:GOTO12
50
1246 IFPOS(0)+LEN(RICHT$(RI))<37 THENPRI
NT", ";RICHT$(RI);:GOTO1250
1247 IFPOS(0)+LEN(RICHT$(RI))>37THENPRIN
T:PRINTRICHT$(RI);:GOTO1250
1249 IFPOS(0)<16 AND POS(0)>2 THENPRINT"
, ";RI$(RI);:GOTO1250
1250 NEXT RICHTUNG
1260 PRINT"."
1270 PRINT"▮────────────────────────────
──────────"
1275 IFSP=25THENGOSUB8050
1280 POKE211,0:POKE214,24:SYS 58732:PRIN
T"▮";:INPUT"WAS SOLL ICH TUN";EI$:PRINT"
▮▮";
1285 IFSP=17THENGOSUB9040
1286 IFSP=23THENGOSUB9080
1287 IFSP=15THENGOSUB9100
1290 IFEI$="N"ANDDU(SP,1)<>0THENSP=DU(SP
,1):PRINT"O.K":GOTO1080
```

```
1300 IFEI$="S"ANDDU(SP,2)<>0THENSP=DU(SP
,2):PRINT"O.K":GOTO1080
1310 IFEI$="W"ANDDU(SP,3)<>0THENSP=DU(SP
,3):PRINT"O.K":GOTO1080
1320 IFEI$="O"ANDDU(SP,4)<>0THENSP=DU(SP
,4):PRINT"O.K":GOTO1080
1330 IFEI$="OB"ANDDU(SP,5)<>0THENSP=DU(S
P,5):PRINT"O.K":GOTO1080
1340 IFEI$="U"ANDDU(SP,6)<>0THENSP=DU(SP
,6):PRINT"O.K":GOTO1080
1350 IF LEN(EI$)<3 THEN PRINT"DAHIN FUER
HT KEIN WEG !":GOTO 1080
1360 IF LEFT$(EI$,3)<>"INV"THENGOTO1990
1370 PRINT"ICH TRAGE FOLGENDES MIT MIR:"
1380 FORI=1TOAO:IFOB(I)=-1THENPRINTOB$(I
):NEXTI
1400 GOTO1080
1990 IFSP=15THENGOSUB9100
2000 LN=LEN(EI$)
2010 FOR EL=1 TO LN:TE$=MID$(EI$,EL,1)
2030 IF TE$<>" "THEN NEXT EL
2040 EV$=LEFT$(EI$,3):RL=LN-EL:IFRL<0THE
N2070
2065 EO$=RIGHT$(EI$,RL)
2066 EO$=LEFT$(EO$,3)
2070 FOR VN=1 TO AV
2080 IF EV$=VE$(VN) THEN 2110
2090 NEXT VN
2100 PRINT"DAS VERB VERSTEHE ICH NICHT!"
:GOTO 1080
2110 FOR N=1 TO AO
2120 IF EO$=RN$(N) THEN 4000
2130 NEXT N
2140 PRINT"ICH VERSTEHE DAS OBJEKT NICHT
!":GOTO 1080
4000 ON VN GOTO 5000,5500,5600,5700,5800
,5900,6000,6100
5000 IF OB(N)<>SP AND OB(N)<>-1 THEN GOT
O 5080
5001 IFN=16ANDSP=20ANDOB(31)<>-1ANDOB(40
)<>-1THENPRINTMS$(7):OB(31)=20:OB(40)=20
```

```
:GOTO1080
5002 IF N=10 AND SP=8 THEN PRINT MS$(3):
GOTO1080
5003 IF N=11 AND SP=7 THEN PRINT MS$(3):
GOTO1080
5004 IF N=4 AND(SP=3 OR OB(N)=-1) THEN P
RINT MS$(1):GOTO1080
5005 IF N=5 AND(SP=3 OR OB(N)=-1) THEN P
RINT MS$(1):GOTO1080
5006 IF N=1 AND SP=1 THEN PRINT MS$(1):G
OTO1080
5008 IF N=3 AND SP=2ANDOB(12)=0THENPRINT
MS$(8):OB(12)=2:GOTO1080
5009 IFN=3ANDSP=2THENPRINTMS$(1):GOTO108
0
5010 IFN=6ANDOB(5)=0THENPRINT"AUF DEM WA
SSER TREIBT EINE FLASCHE.":OB(5)=SP:GOTO
1080
5011 IFN=7AND(SP=1OROB(N)=-1)THENPRINT"E
S IST EIN BUCH UEBER ZAUBERTRAENKE.":GOT
O1080
5012 IFN=8ANDSP=5ANDOB(16)=0THENPRINTMS$
(9):OB(16)=SP:GOTO1080
5013 IF N=9 AND NOT FL(1) THEN PRINTMS$(
1):GOTO1080
5014 IF N=9 AND FL(1)ANDSP=5 THENPRINT"E
R IST NICHT FUNKTIONSFAEHIG.":GOTO1080
5015 IFN=12ANDOB(12)=-1THENPRINT"DARAUF
STEHT ETWAS GESCHRIEBEN.":OB(16)=SP:GOTO
1080
5016 IFN=12ANDOB(12)=2THENPRINT"DAZU MUS
S ICH IHN ERST NEHMEN.":GOTO1080
5017 IFN=13ANDOB(14)=0THENPRINT"AUF DEM
TISCH LIEGT EIN MESSER.":OB(14)=3:GOTO10
80
5018 IFN=2AND(OB(7)=1OROB(7)=0)THENPRINT
"DARAUF LIEGT EIN ALTES BUCH.":OB(7)=1:G
OTO1080
5019 IFN=14 AND (OB(14)=SP OR OB(14)=-1)
THENPRINT"DIE KLINGE IST SEHR SCHARF.":G
OTO1080
```

```
5020 IFN=15AND(OB(N)=SPOROB(N)=-1)THENPR
INTMS$(1):GOTO1080
5021 IFN=16AND(OB(16)=-1OROB(16)=SP)THEN
PRINTMS$(1):GOTO1080
5022 IFN=17ANDSP=6ANDFL(3)<>-1THENPRINT"
ES IST VERSCHLOSSEN.":GOTO1080
5023 IFN=17ANDSP=6ANDFL(3)=-1THENPRINT"E
S STEHT WEIT OFFEN.":GOTO1080
5024 IFN=18ANDSP=7ANDFL(1)THENPRINT"DIE
ZUGBRUECKE FAELLT NACH UNTEN.":DU(7,4)=1
0:GOTO1080
5025 IFN=18ANDSP=7ANDNOTFL(1)THENPRINT"D
IE ZUGBRUECKE IST HOCHGEZOGEN.":GOTO1080
5026 IFN=21THENPRINTMS$(1):GOTO1080
5027 IFN=23ANDSP=14GOTO10000
5028 IFN=28ANDSP=17THENPRINT"ES IST WIRK
LICH EIN PRACHTEXEMPLAR.":GOSUB9040:GOTO
1080
5029 IFN=29ANDSP=18THENPRINT"EIN MERKWUE
RDIGER GERUCH DRINGT HERAUS.":GOTO1080
5030 IFN=30ANDSP=19OROB(30)=-1THENPRINT"
ES IST WERTLOSES METALL.":GOTO1080
5031 IFN=31AND(SP=20OROB(31)=-1)THENPRIN
TMS$(1):GOTO1080
5032 IFN=35AND(SP=26OROB(35)=-1)THENPRIN
TMS$(1):GOTO1080
5033 IFN=36ANDSP=23THENPRINT"ER SIEHT NI
CHT SEHR TRAGFEST AUS.":GOTO1080
5034 IFN=37ANDSP=24THENPRINT"ES IST DAS
BERUEHMTE BLAUE WASSER.":GOTO1080
5035 IFN=38ANDSP=10THENPRINTMS$(1):GOTO1
080
5036 IFN=40AND(SP=20OROB(40)=-1)THENPRIN
TMS$(1):GOTO1080
5037 IFN=43THENSP=26:PRINT"O.K.":GOTO108
0
5038 IFN=45THENPRINTMS$(10):PRINTMS$(11)
:GOTO1080
5080 IFN=21THENPRINTMS$(1):GOTO1080
5081 IFN=16ANDSP=20ANDOB(31)<>-1ANDOB(40
)<>-1THENPRINTMS$(7):OB(31)=20:OB(40)=20
```

```
:GOTO1080
5082 IFN=33ANDSP=23THENGOSUB9080:PRINT"I
M OSTEN SCHEINT EINE QUELLE ZU SEIN.":GO
TO1080
5083 IFN=16AND(SP=23OROB(34)=-1)THENPRIN
T"ES IST HEILSCHLAMM.":GOSUB9080:GOTO108
0
5098 PRINTMS$(1):GOTO1080
5500 IFOB(N)<>SPANDOB(N)<>-1THENGOTO5580
5501 IFN=1THENPRINTMS$(2):GOTO1080
5502 IFN=6THENPRINTMS$(2):GOTO1080
5503 IFN=17THENPRINTMS$(2):GOTO1080
5504 IFN=2 THENPRINTMS$(4):GOTO1080
5505 IFN=3 THENPRINTMS$(4):GOTO1080
5506 IFN=8 THENPRINTMS$(4):GOTO1080
5507 IFN=10THENPRINTMS$(4):GOTO1080
5508 IFN=11THENPRINTMS$(4):GOTO1080
5509 ZA=0:FORI=1TOAO:IFOB(I)=-1THENZA=ZA
+1:IFZA=4THENGOTO5599
5510 NEXTI
5511 IFN=5THENOB(N)=-1:PRINT"O.K.":GOTO1
080
5512 IFN=7THENOB(N)=-1:PRINT"O.K.":GOTO1
080
5513 IFN=12THENOB(N)=-1:PRINT"O.K.":GOTO
1080
5514 IFN=15THENOB(N)=-1:PRINT"O.K.":GOTO
1080
5515 IFN=16THENOB(N)=-1:PRINT"O.K.":GOTO
1080
5516 IFN=14THENOB(N)=-1:PRINT"O.K.":GOTO
1080
5517 IFN=9THENPRINTMS$(2):GOTO1080
5518 IFN=23THENPRINTMS$(2):GOTO1080
5519 IFN=28THENPRINT"WOMIT SOLL ICH SIE
FANGEN ?":GOSUB9040:GOTO1080
5520 IFN=18THENPRINTMS$(0):GOTO1080
5521 IFN=30THENOB(N)=-1:PRINT"O.K.":GOTO
1080
5522 IFN=31THENOB(31)=-1:PRINT"ICH HABE
DIE RUESTUNG ANGELEGT.":GOTO1080
```

```
5523 IFN=35ANDOB(31)=-1THENPRINT"O.K.":O
B(35)=-1:GOTO1080
5524 IFN=35ANDOB(31)<>-1THENGOTO8000
5525 IFN=36THENPRINTMS$(4):GOTO1080
5526 IFN=37ANDOB(5)=-1THENPRINT"O.K.":OB
(5)=0:OB(41)=-1:GOTO1080
5527 IFN=37ANDOB(5)<>-1THENPRINT"ZUVOR B
RAUCHE ICH EINE LEERE FLASCHE.":GOTO1080
5528 IFN=38THENPRINTMS$(0):GOTO1080
5529 IFN=40THENOB(40)=-1:PRINT"O.K.":GOT
O1080
5532 IFN=13THENPRINTMS$(4):GOTO1080
5533 IFN=4THENOB(N)=-1:PRINT"O.K.":GOTO1
080
5580 IFN=16ANDOB(4)<>-1THENPRINT"ZUVOR B
RAUCHE ICH EIN GEFAESS.":GOTO1080
5581 IFN=16ANDOB(4)=-1THENPRINT"O.K.":OB
(34)=-1:GOTO1080
5589 PRINT"SO ETWAS SEHE ICH HIER NICHT.
":GOTO1080
5590 GOTO1080
5599 PRINT"TUT MIR LEID, ICH TRAGE SCHON
 GENUG !":GOTO1080
5600 IFN=12ANDOB(12)<>-1THENPRINT"ICH HA
BE KEINEN ZETTEL.":GOTO1080
5601 IFN=12ANDOB(12)=-1THENCO$=RIGHT$(ST
R$(TI),3):CO=VAL(CO$):PRINTMS$(5);CO:GOT
O1080
5602 IFN=7ANDOB(7)=-1THENGOTO5650
5603 IFN=7ANDOB(7)<>-1THENPRINT"DAZU MUS
S ICH ES ERST HABEN.":GOTO1080
5649 GOTO1080
5650 INPUT"WELCHE SEITE";SE
5651 IF SE<>COTHENPRINTMS$(1):GOTO1080
5652 PRINT"♥":PRINTSPC(2)"RATSCHLAEGE FU
ER BESONDERE NOTLAGEN.":PRINT
5654 PRINT"SOLLTE ES DIR ANGESCHEHEN, DA
SS DIE      DEINEN VON EINEM ZAUBERER
5655 PRINT"BOESER GESINNUNG IN DEN LANGE
N SCHLAF    VERSETZT WERDEN, SO HOERE MEI
NEN"
```

```
5656 PRINT"RAT:
5657 PRINT:PRINT:PRINT:PRINT"VERSCHAFFE
DIR DIE ZUTATEN ZU HUMBUGS"
5658 PRINT"HEILWASSER UND BRINGE SIE DEM
 ZAUBERER"
5659 PRINT"JENSEITS DES ZAUBERWALDES."
5660 PRINT"SOLLTE ER DIR DANN WOHL GESON
NEN SEIN,  SO WIRD ER DIR HELFEN,"
5661 PRINT"WENN NICHT, KANNST DU NUR NOC
H AUF    EINEN PRINZEN HOFFEN, DER"
5662 PRINT"DEINE TOCHTER UND MIT IHR ALL
E"
5663 PRINT"BEWOHNER DES SCHLOSSES WACHKU
ESST."
5665 GETA$:IFA$=""THEN5665
5666 PRINT"□":GOTO1080
5678 PRINT"WEBEN, AMEISENEIERN SOWIE SCH
LANGENEIERNUND BRUEHE DIESES MIT DEM WUN
DER-"
5679 PRINT"TAETIGEM BLAUEN WASSER AUF."
5700 IFN=17ANDOB(16)<>-1THENPRINT"WOMIT
?":GOTO1080
5701 IFN=17ANDOB(16)=-1THENPRINT"O.K.":D
U(6,4)=7:FL(3)=-1:GOTO1080
5702 IFN=35ANDOB(35)THENPRINT"DARIN WAR
EINE ZAEHE FLUESSIGKEIT.":GOTO1080
5799 GOTO1080
5800 IFN=9ANDOB(14)=-1ANDSP=5THENPRINT"D
IE HALTESEILE SIND ZERSCHNITTEN.":FL(1)=
-1:GOTO1080
5801 IFN=9ANDSP=5ANDOB(14)<>-1THENPRINT"
WOMIT ?":GOTO1080
5899 PRINTMS$(0):GOTO1080
5900 IFOB(N)<>SPANDOB(N)<>-1THENPRINT"SO
 ETWAS SEHE ICH HIER NICHT !":GOTO1080
5901 IFN=9THENPRINT"DAZU SIND MINDESTENS
 ZWEI MANN NOETIG.":GOTO1080
5902 IFOB(14)=-1ANDSP=5THENPRINT"DIE HAL
TESEILE SIND ZERSCHNITTEN.":FL(1)=-1:GOT
O1080
5903 IFN=31THENPRINT"O.K. - ICH BIN IN D
```

```
ER RUESTUNG.":OB(N)=-1:GOTO1080
6000 IFN=5ANDSP=24ANDOB(5)=-1THENPRINT"O
.K.":OB(5)=0:OB(41)=-1:GOTO1080
6010 IFN=5ANDOB(5)=-1ANDSP<>24THENPRINT"
WOMIT ?":GOTO1080
6100 IFOB(N)<>-1THENGOTO6180
6101 IFN=4THENPRINT"O.K.":OB(N)=SP:GOTO1
080
6102 IFN=5THENPRINT"O.K.":OB(N)=SP:GOTO1
080
6103 IFN=7THENPRINT"O.K.":OB(N)=SP:GOTO1
080
6104 IFN=12THENPRINT"O.K.":OB(N)=SP:GOTO
1080
6105 IFN=15THENPRINT"O.K.":OB(N)=SP:GOTO
1080
6106 IFN=16THENPRINT"O.K.":OB(N)=SP:GOTO
1080
6107 IFN=14THENPRINT"O.K.":OB(N)=SP:GOTO
1080
6108 IFN=30THENPRINT"O.K.":OB(N)=SP:GOTO
1080
6109 IFN=41THENPRINT"O.K.":OB(N)=SP:GOTO
1080
6110 IFN=45THENPRINT"O.K.":OB(N)=SP:GOTO
1080
6111 IFN=40THENPRINT"O.K.":OB(N)=SP:GOTO
1080
6112 IFN=36THENPRINT"O.K.":OB(N)=SP:GOTO
1080
6113 IFN=35THENPRINT"O.K.":OB(N)=SP:GOTO
1080
6114 IFN=31THENPRINT"O.K.":OB(N)=SP:GOTO
1080
6170 GOTO1080
6180 IFN=5ANDOB(41)=-1THENPRINT"O.K.":OB
(41)=SP:GOTO1080
6181 IFN=16ANDOB(34)=-1THENPRINT"O.K.":O
B(34)=SP:GOTO1080
6190 GOTO1080
8000 PRINT"☐":PRINT"EINE SCHLANGE HAT MI
```

```
CH GEBISSEN."
8010 PRINT"ICH BIN TOD.":PRINT:PRINT"SOL
L ICH ES NOCH EINMAL VERSUCHEN ?"
8020 GETA$:IFA$=""THEN8020
8030 IF A$="N"THENEND
8040 RUN
8050 IFOB(34)<>25THENRETURN
8060 IFOB(35)<>25THENRETURN
8070 IFOB(41)<>25THENRETURN
8080 IFOB(40)<>25THENRETURN
8090 PRINT"□":PRINT"■EINE STIMME ERTOENT
:":PRINT:PRINT:PRINT
8100 PRINT"■DER GROSSE HUMBUG FREUT SICH
 UEBER"
8110 PRINT:PRINT"DEINE GABEN. ER WIRD SE
INE ZAUBERKRAFT"
8120 PRINT:PRINT"FUER DICH EINSETZEN.":P
RINT:PRINT
8130 PRINT"GEHE NUN NACH HAUSE, DENN DOR
T WIRD" :PRINT
8140 PRINT"BEREITS EIN FEST VORBEREITET,
 UM DAS ":PRINT
8150 PRINT"ERWACHEN AUS DEM LANGEN SCHLA
F ZU ":PRINT:PRINT"FEIERN."
8160 GOTO8160
8170 PRINT"□
9040 IFOB(30)=-1THENOB(30)=20:PRINTMS$(6
):RETURN
9050 IFOB(35)=-1THENOB(35)=20:PRINTMS$(6
):RETURN
9060 IFOB(5)=-1THENOB(5)=20:PRINTMS$(6):
RETURN
9070 RETURN
9080 IFOB(31)<>-1THENRETURN
9090 PRINT"□":PRINT"ICH BIN ZU SCHWER UN
D VERSINKE !" :GOTO8010
9100 IFRND(1)>.3THENRETURN
9110 PRINT"□": PRINT"ICH BIN IN EINE FAL
LGRUBE GESTUERZT.":GOTO8010
10000 PRINT"□":PRINT"ES IST EINE ZAUBERK
UECHE. DER ZAUBERER"
```

```
10010 PRINT"KOMMT UND MACHT EINEN STEIN
AUS MIR.":GOTO8010

READY.
```

```
0 REM    C64 ADVENTURE EDITOR
1 REM  (C) 1984  BY J. WALKOWIAK
2 REM-----------------------------------
5 POKE 53280,0: POKE 53281,0: PRINT"■"
10 DIM RAUM$(60),OB$(60),RN$(60),OB(60),
VE$(30),MS$(60),AC$(30,60),BC$(30,60)
15 DIM AD$(20),DU(60,6)
16 M1$="        ADVENTURE EDITOR   VER 1.
0"
17 M2$="■■■WAS WUENSCHEN SIE"
18 M3$="────────────────────────────────
───────"
19 M4$="
       "
20 AR=0:AO=0:AV=0:AM=0:AF=0:I2=1:L3=1
30 PRINT"■";M1$:PRINTM3$
40 PRINT"■■WELCHES ADVENTURE WOLLEN SIE
BEARBEITEN":INPUT NA$:INPUT"■VERSION ";V
E$
50 INPUT"■COPYRIGHT BY ";CR$
200 PRINT"■■";M1$:PRINT M3$;
210 PRINTSPC(10)"■0 - DATEN"
211 PRINTTAB(10)"■1 - RAEUME EINGEBEN"
212 PRINTTAB(10)"■2 - OBJEKTE EINGEBEN"
213 PRINTTAB(10)"■3 - VERBEN EINGEBEN"
214 PRINTTAB(10)"■4 - STARTPOSITIONEN"
215 PRINTTAB(10)"■5 - RAEUME VERBINDEN"
216 PRINTTAB(10)"■6 - BEDINGUNGEN & AKTI
ONEN"
217 PRINTTAB(10)"■7 - MELDUNGEN EINGEBEN
"
218 PRINTTAB(10)"■8 - DISKETTENZUGRIFF"
219 PRINTTAB(10)"■9 - PROGRAMMENDE■"
220 PRINTM3$
225 PRINTTAB(10)"BITTE WAEHLEN SIE"
230 GOSUB10000
235 IF A=9 THEN EN=-1:GOTO 5100
240 ON A+1 GOTO 3000,1100,1200,1300,1400
,1500,1600,4000,5000,6000
```

```
499 REM ------ UNTERPROGRAMM EIN/AUSGABE
500 PRINT"□";
510 ZE=0:SP=0:GOSUB 11000:PRINT T1$
520 Z=Z+1
555 ZE=0:SP=25:GOSUB 11000:PRINTT2$;Z
556 PRINTM3$
557 ZE=16:SP=0:GOSUB 11000:PRINT M4$
558 ZE=16:SP=0:GOSUB 11000:PRINTZ-1;:IF
A=1 THEN PRINT RAUM$(Z-1)
559 IF A=2 THEN PRINT OB$(Z-1)
560 IF A=3 THEN PRINT VERB$(Z-1)
563 ZE=20:SP=0:GOSUB 11000:PRINTT3$;:EI$
="":INPUT EINGABE$
564 IF LEN(EI$)=0 THEN GOTO 200
565 IF A=2 THEN ZE=22:SP=0:GOSUB 11000:I
NPUT"RUFNAME  ";RN$(Z)
570 ZE=20:SP=0:GOSUB 11000:PRINT M4$
571 ZE=22:SP=0:GOSUB 11000:PRINT M4$
590 RETURN
1100 PRINT"□":T1$="ORTE EINGEBEN ":T2$="
 RAUM NR.:":T3$="ICH BIN "
1105 Z=AR
1110 GOSUB 510      :REM RAEUME DRUCKEN
1120 RAUM$(Z)=EINGABE$
1125 AR=Z
1130 GOTO 1110
1199 REM ------------- EINGABE ORTE ENDE
1200 PRINT"□":T1$="OBJEKTE EINGEBEN ":T2
$=" OBJEKT NR":T3$="ICH SEHE "
1205 Z=AO
1210 GOSUB 510
1220 OB$(Z)=EINGABE$
1225 AO=Z
1230 GOTO1210
1290 REM ---------- ENDE EINGABE OBJEKTE
1300 PRINT"□"
1301 T1$="VERBEN EINGEBEN":T2$="VERB NR.
:"
1302 T3$=""
1305 Z=AV
1310 GOSUB 510
```

```
1320 VERB$(Z)=EINGABE$
1330 AV=Z
1360 GOTO 1310
1390 REM ---------- ENDE EINGABE OBJEKTE
1399 REM ------------ OBJEKTE PLAZIEREN
1400 PRINT"□"
1410 LP=L3
1420 FOR I=LP TO AO
1430 GOSUB 12000      :REM LISTE DRUCKEN
1440 ZE=23:SP=0:GOSUB11000:PRINT"OBJEKT
▒";RN$(I);" ▔IN RAUM NR.";:INPUT OB(I)
1450 LP=I+1
1460 PRINT"□"
1470 NEXT I
1490 GOTO 200
1500 FOR R1=I2 TO AR
1510 PRINT"□"
1520 GOSUB 12000
1530 PRINT"▒RAUM";R1;"FUEHRT IM NORDEN N
ACH";:INPUT DU(R1,1)
1540 PRINT"▒RAUM";R1;"FUEHRT IM SUEDEN N
ACH";:INPUT DU(R1,2)
1550 PRINT"▒RAUM";R1;"FUEHRT IM WESTEN N
ACH";:INPUT DU(R1,3)
1560 PRINT"▒RAUM";R1;"FUEHRT IM OSTEN  N
ACH";:INPUT DU(R1,4)
1570 PRINT"▒RAUM";R1;"FUEHRT NACH OBEN N
ACH";:INPUT DU(R1,5)
1580 PRINT"▒RAUM";R1;"FUEHRT NACH UNTEN
NACH";:INPUT DU(R1,6)
1585 NEXT R1
1590 GOTO200
1591 REM ------------ ENDE VERBINDUNGEN
1600?"□": ------------ REM BEDINGUNGEN
1600 REM ------- BEDINGUNGEN & AKTIONEN
1605 NB=0
1607 NB=NB+1
1608 IF NB>AV THEN GOTO 200
1609 RN$(0)="<WEITER>"
1610 PRINT"□":FORI=0 TO AO
1620 PRINT I;RN$(I);"|";
```

```
1630 IF POS(0)+LEN(RN$(I+1)) > 38 THEN P
RINT
1640 NEXT I
1650 PRINT: PRINT M3$:PRINT"█"
1700 PRINT"AKTION AUF VERB █";VERB$(NB);
" █UND OBJEKT NR.";
1710 INPUT OB
1715 IF OB=0 THEN GOTO 1607
1800 PRINT"█";
1810 PRINT"BITTE ALLE BEDINGUNGEN FUER D
IE AKTION  █";VE$(NB);" ";RN$(OB)";█ EIN
GEBEN
1815 PRINTM3$;
1820 PRINT"██"TAB(5)"R    - OBJEKT IST IM
 RAUM"
1830 PRINTTAB(5)"█I    - OBJEKT IST IM IN
VENTORY"
1840 PRINTTAB(5)"█N    - OBJEKT IST NICHT
 VORHANDEN"
1850 PRINTTAB(5)"█FX   - FLAG X IST GESET
ZT"
1860 PRINTTAB(5)"█GX   - FLAG X IST GELOE
SCHT"
1870 PRINTTAB(5)"█SXX  - SPIELER IST IM R
AUM XX "
1880 ZE=20:SP=0:GOSUB11000
1890 PRINT M3$;:PRINT"ALTER CODE ===> ";
BC$(NB,OB):PRINT
1900 INPUT"BEDINGUNGEN";BC$(NB,OB)
1905 PRINT"██";VE$(NB);" ";OB$(OB);" BEW
IRKT, WENN:"
1906 PRINTBC$(NB,OB);"█ ERFUELLT, FOLGEN
DE AKTION:":PRINTM3$;
1910 PRINT"     V    - ";RN$(OB);" VERSCH
WINDET"
1920 PRINTTAB(5)"█I    - ";RN$(OB);" KOMM
T INS INV"
1930 PRINTTAB(5)"█NXX  - OBJEKT XX ERSCHE
INT NEU"
1940 PRINTTAB(5)"█DXY  - DURCHGANG N. RAU
M X
```

```
1950 PRINTTAB(5)"■SXX - SPIELER NACH RAU
M XX "
1960 PRINTTAB(5)"■FX  - FLAG X SETZEN"
1970 PRINTTAB(5)"■LX  - FLAG X LOESCHEN"
1980 PRINTTAB(5)"■MXX - MELDUNG XX AUSGE
BEN"
1990 PRINTTAB(5)"■T   - SPIELER STIRBT"
2000 PRINTTAB(5)"■E   - ENDE, DA GEWONNE
N"
2005 PRINTM3$;
2010 PRINT"ALTER CODE ===> ";AC$(NB,OB)
2020 INPUT"AKTION";AC$(NB,OB)
2030 GOTO1610
2040 REM ------ ENDE AKTION & BEDINGUNG
2080 GOSUB10000
2999 REM -------------------- KENNDATEN
3000 PRINT"□KENNDATEN DES ADVENTURES ■";
NA$
3010 PRINT"■"M3$"■"
3020 PRINT" RAEUME:  ";AR
3030 PRINT" OBJEKTE:";AO
3040 PRINT" VERBEN:  ";AV
3050 PRINT" BEDING.:";NB
3060 PRINT" MELDG.:  ";AM;"■"
3070 IF LO THEN PRINT" ";AF;"BENUTZTE FL
AGS"
3080 PRINTM3$
3090 PRINT"STARTRAUM DES SPIELERS:
";SP
3095 PRINT"■ERFORDERLICHE EINGABELAENGE:
 ";WL
3100 GOSUB10000
3110 GOTO200
3999 REM ------------------ MITTEILUNGEN
4000 PRINT"□MITTEILUNGEN EINGEBEN"
4010 PRINTM3$
4020 AM=AM+1
4030 PRINT AM;:INPUT MS$(AM)
4040 IF LEN(EI$)=0 THEN AM=AM-1 : GOTO 2
00
4050 GOTO 4020
```

```
4060 REM ------------- ENDE MITTEILUNGEN
4999 REM -------------- DISKETTENBEFEHLE
5000 PRINT"□1 - SPEICHERN"
5010 PRINT"▣2 - LADEN"
5020 GOSUB 10000
5030 IF A<1 OR A>2 THEN GOTO 5020
5040 ON A GOTO 5100, 5500
5100 PRINT"□BITTE EINEN MOMENT WARTEN ..
.";:PRINT"▣SPEICHERUNG LAEUFT !"
5105 GOSUB6000
5110 OPEN 2,8,2,"§0:"+NA$+",S,W"
5120 PRINT#2,NA$:PRINT#2,VERS$:PRINT#2,C
R$:PRINT#2,WL
5130 PRINT#2,AR:PRINT#2,AO:PRINT#2,AV:PR
INT#2,AM:PRINT#2,SP:PRINT#2,AF
5140 FOR I=1 TO AR:PRINT#2,RA$(I):NEXT I
5150 FOR I=1 TO AO:PRINT#2,OB$(I):PRINT#
2,RN$(I):PRINT#2,OB(I):NEXT I
5170 FOR I=1 TO AV:PRINT#2,VE$(I):NEXT I
5180 FOR I=1 TO AM:PRINT#2,MS$(I):NEXT I
5190 FOR X=1 TO AR
5200 FOR Y=1 TO 6
5210 PRINT#2,DU(X,Y)
5220 NEXT Y
5230 NEXT X
5240 FOR X=1 TO AV
5250 FOR Y=1 TO AO
5260 PRINT#2,BC$(X,Y)
5270 NEXT Y
5280 NEXT X
5290 FOR X=1 TO AV
5300 FOR Y=1 TO AO
5310 PRINT#2,AC$(X,Y)
5320 NEXT Y
5330 NEXT X
5340 CLOSE 2 : IF EN THEN PRINT"□":END
5350 GOTO 200
5500 PRINT"□BITTE EINEN MOMENT WARTEN ..
.":LO=-1 :PRINT"▣DATEN WERDEN GELADEN !"
5510 OPEN 2,8,2,NA$+",S,R"
5520 INPUT#2,NA$:INPUT#2,VERS$:INPUT#2,C
```

```
R$:INPUT#2,WL
5530 INPUT#2,AR:INPUT#2,AO:INPUT#2,AV:IN
PUT#2,AM:INPUT#2,SP:INPUT#2,AF
5540 FOR I=1 TO AR:INPUT#2,RA$(I):NEXT I
5550 FOR I=1 TO AO:INPUT#2,OB$(I):INPUT#
2,RN$(I):INPUT#2,OB(I):NEXT I
5570 FOR I=1 TO AV:INPUT#2,VE$(I):NEXT I
5580 FOR I=1 TO AM:INPUT#2,MS$(I):NEXT I
5590 FOR X=1 TO AR
5600 FOR Y=1 TO 6
5610 INPUT#2,DU(X,Y)
5620 NEXT Y
5630 NEXT X
5640 FOR X=1 TO AV
5650 FOR Y=1 TO AO
5660 INPUT#2,BC$(X,Y)
5670 NEXT Y
5680 NEXT X
5690 FOR X=1 TO AV
5700 FOR Y=1 TO AO
5710 INPUT#2,AC$(X,Y)
5720 NEXT Y
5730 NEXT X
5740 CLOSE 2
5750 GOTO 200
5999 REM ----------------- UEBRIGE DATEN
6000 PRINT"□BITTE GEBEN SIE DIE NOCH NOT
WENDIGEN    DATEN EIN:":PRINTM3$
6010 INPUT"▓▓WORTLAENGE";WL
6020 INPUT"▓▓STARTRAUM ";SP
6030 INPUT"▓▓ANZAHL FLAGS";AF
6090 RETURN
10000 GET A$:IFA$="" THEN 10000
10010 A=VAL(A$):RETURN
11000 POKE 211,SPALTE:POKE 214,ZEILE:SYS
 58732:RETURN
12000 SP=0:ZE=0:GOSUB11000
12010 PRINT"LISTE DER MOEGLICHEN RAEUME:
":PRINTM3$
12020 FOR I1=1 TO AR:REM
                         RAUMLISTE DRUCKEN
```

```
12030 PRINTI1;RAUM$(I1);
12040 IF POS(0)+LEN(RAUM$(I1+1)) >38 THE
N PRINT
12050 NEXT I1
12060 PRINT"▒"
12070 RETURN

READY.
```

```
0 REM -- ADVENTURE INTERPRETER VER 1.0
1 REM    (C) 1984  BY JOERG WALKOWIAK
2 REM -----------------------------------
-
5 POKE 53280,0:POKE 53281,0
6 PRINT"{clr}{down}{down}{down}{down}{down}{down}{down}{down}"SPC(10)"ADVENTURE SYSTE
M  1.0"
7 PRINT"{down}              (C) 1984 BY WALKOWIAK
"
8 PRINT"{down}{down}{down}{down}{rvs}               AUS DEM DATA BECKE
R BUCH"
9 PRINTSPC(16)"{wht}ADVENTURES":PRINT"
 UND WIE MAN SIE PROGRAMMIERT"
30 FORI=1TO2000:NEXTI
40 PRINT"{clr}{wht}WELCHES ADVENTURE WOLLEN SIE
SPIELEN":INPUT"{down}{down}DATEINAME:  ";NA$
50 OPEN 2,8,2,NA$+",S,R"
60 INPUT#2,NA$:PRINTNA$:INPUT#2,VERS$:PR
INTVERS$:INPUT#2,CR$:PRINTCR$:INPUT#2,WL
70 INPUT#2,AR:INPUT#2,AO:INPUT#2,AV:INPU
T#2,AM:INPUT#2,SP:INPUT#2,AF
80 FOR I=1 TO AR:INPUT#2,RA$(I):NEXTI
90 FORI=1TOAO:INPUT#2,OB$(I):INPUT#2,RN$
(I):RN$(I)=LEFT$(RN$(I),WL):INPUT#2,OB(I
):NEXTI
100 FOR I=1 TO AV:INPUT#2,VE$(I):VE$(I)=
LEFT$(VE$(I),WL): NEXT I
110 FOR I=1 TO AM:INPUT#2,MS$(I):NEXTI
120 FOR X=1 TO AR : FOR Y=1 TO 6
130 INPUT#2,DU(X,Y)
140 NEXT Y : NEXT X
150 FOR X=1 TO AV : FOR Y=1 TO AO
160 INPUT#2,BC$(X,Y)
170 NEXT Y : NEXT X
180 FOR X=1 TO AV : FOR Y=1 TO AO
190 INPUT#2,AC$(X,Y)
200 NEXT Y : NEXT X
210 CLOSE2
1000 PRINT"{clr}":PRINT CHR$(142)
```

```
1010 LE$="
       "
1020 DATA NORDEN,SUEDEN,OSTEN,WESTEN,OBE
N,UNTEN
1030 FOR RI=1 TO 6
1040 READ RI$(RI)
1050 NEXT RI
1070 PRINT"□":POKE 53280,0:POKE53281,0
1080 PRINT"□"
1090 POKE 211,0:POKE 214,0: SYS 58732
1100 FOR ZE=1 TO 10
1110 PRINT LE$
1120 NEXT ZE
1130 POKE 211,0:POKE 214,0: SYS 58732
1140 PRINT"ICH BIN ";
1150 PRINTRA$(SP)
1160 PRINT"ICH SEHE ";:GE=0
1170 FOR I=1 TO AO
1180 IF OB(I)<>SP THEN 1210
1190 IF POS(0)+LEN(OB$(I))+2 < 39 THEN P
RINT OB$(I);", ";:GE=-1:GOTO 1210
1200 IF POS(0)+LEN(OB$(I))+2 >= 39 THEN
PRINT : GOTO 1190
1210 NEXT I : IF NOT GE THEN PRINT"NICHT
S BESONDERES  ";
1220 PRINT"■■L."
1230 PRINT LE$
1240 PRINT"□ICH KANN NACH ";
1250 FOR RI=1 TO 6
1260 IF DU(SP,RI)=0 THEN GOTO 1310
1270 IF POS(0)=14 THEN PRINT RI$(RI);:GO
TO 1310
1280 IF POS(0)+LEN(RI$(RI))<37 THEN PRIN
T", ";RI$(RI);:GOTO 1310
1290 IF POS(0)+LEN(RI$(RI))>=37 THEN PRI
NT", ":PRINTRI$(RI);:GOTO 1310
1300 IF POS(0)<16 AND POS(0)>2 THEN PRIN
T", ";RI$(RI);
1310 NEXT RI
1320 PRINT"."
1330 PRINT"■────────────────────────────────
```

```
________"
1390 POKE 211,0:POKE 214,24:SYS 58732:PR
INT"■";:INPUT"WAS SOLL ICH TUN";EI$:PRIN
T"■■";
1425 IF LEN(EI$)>2 THEN 1500
1430 IF EI$="N"ANDDU(SP,1)<>0 THEN SP=DU
(SP,1):PRINT"O.K.":GOTO 1080
1440 IF EI$="S"ANDDU(SP,2)<>0 THEN SP=DU
(SP,2):PRINT"O.K.":GOTO 1080
1450 IF EI$="W"ANDDU(SP,3)<>0 THEN SP=DU
(SP,3):PRINT"O.K.":GOTO 1080
1460 IF EI$="O"ANDDU(SP,4)<>0 THEN SP=DU
(SP,4):PRINT"O.K.":GOTO 1080
1470 IF EI$="OB"ANDDU(SP,5)<>0 THEN SP=D
U(SP,5):PRINT"O.K.":GOTO 1080
1480 IF EI$="U"ANDDU(SP,6)<>0 THEN SP=DU
(SP,6):PRINT"O.K.":GOTO 1080
1490 PRINT"DAHIN FUHRT KEIN WEG !":GOTO1
080
1500 IF LEFT$(EINGABE$,3)<>"INV"THEN 160
0
1510 PRINT"ICH TRAGE FOLGENDES MIT MIR:"
1520 FOR I=1 TO AO
1530 IF OB(I)=-1 THEN PRINT OB$(I)
1540 NEXT I
1550 GOTO 1080
1600 IF LEFT$(EI$,4)<>"SAVE"THEN 1700
1605 PRINT"□"SPC(10)"■SPIELSTAND SPEICHE
RN■"
1606 INPUT"■■UNTER WELCHEM NAMEN ";EI$
1610 IF LEN(EI$)>16 THEN 1605
1615 PRINT"■■FILE ";EI$;" WIRD GESCHRIEB
EN."
1620 OPEN 2,8,2,"$0:"+EI$+",S,W"
1625 PRINT#2,NA$:PRINT#2,SP:PRINT#2,FL
1630 FOR I=1 TO AO:PRINT#2,OB(I):NEXTI
1640 FOR RA=1 TO AR:FOR RI=1 TO 6
1650 PRINT#2,DU(RA,RI)
1660 NEXT RI: NEXT RA
1670 FOR I=1 TO AF: PRINT#2,FL(I):NEXT I
1675 CLOSE2:GOSUB 1680:PRINT"□":GOTO1080
```

```
1680 OPEN1,8,15:INPUT#1,A,B$,C,D
1685 IFA<>0THENPRINT:PRINT"ACHTUNG:
":PRINTB$:FORI=1TO5000:NEXT:CLOSE2:CLOSE
1:GOTO1080
1690 CLOSE1:RETURN
1700 IF LEFT$(EI$,4)<>"LOAD"THEN 2000
1705 PRINT""SPC(10)"SPIELSTAND LADEN"
1706 INPUT"WELCHES SPIEL";EI$
1710 IF LEN(EI$)>16 THEN 1705
1715 PRINT"FILE ";EI$;" WIRD GELADEN"
1720 OPEN 2,8,2,"$0:"+EI$+",S,R"
1725 INPUT#2,NA$:INPUT#2,SP:INPUT#2,FL
1730 FOR I=1 TO AO:INPUT#2,OB(I):NEXTI
1740 FOR RA=1 TO AR:FOR RI=1 TO 6
1750 INPUT#2,DU(RA,RI)
1760 NEXT RI: NEXT RA
1770 FOR I=1 TO AF: INPUT#2,FL(I):NEXT I
1775 CLOSE2:GOSUB 1680:PRINT"":GOTO1080
2000 LN=LEN(EI$)
2010 FOR EL=1 TO LN
2020 TE$=MID$(EI$,EL,1)
2030 IF TE$<>" "THEN NEXT EL
2040 EV$=LEFT$(EI$,WL)
2050 RL=LN-EL
2060 IF RL<0 THEN 2090
2070 EO$=RIGHT$(EI$,RL):EO$=LEFT$(EO$,WL
)
2090 FOR VN=1 TO AV
2100 IF EV$=VE$(VN) THEN 2130
2110 NEXT VN
2120 PRINT"DAS VERB VERSTEHE ICH NICHT."
:GOTO1080
2130 FOR N=1 TO AO
2140 IF EO$=RN$(N) THEN 2200
2150 NEXT N
2160 PRINT"ICH VERSTEHE DAS OBJEKT NICHT
.":GOTO1080
2198 REM -------------------------------
------- VN UND N SIN VERB/OBJEKT NUMMERN
2199 REM ---------- BEDINGUNGEN ERFUELLT
2200 FOR AB=1 TO LEN (BC$(VN,N))
```

```
2210 BD$(AB)=MID$(BC$(VN,N),AB,1)
2220 NEXT AB
2250 FOR AA=1 TO LEN(AC$(VN,N))
2260 AD$(AA)=MID$(AC$(VN,N),AA,1)
2270 NEXT AA
2280 REM --- BEDINGUNGEN IN BD$
             AKTIONEN IN AD$ ----------
2290 AB=AB-1:AA=AA-1
2300 X=0:ER=0
2310 X=X+1
2320 IF X=AB+1 THEN 2500: REM ---------
--------- ALLE BEDINGUNGEN UEBERPRUEFT --
---
2330 IF BD$(X)<>"R"THEN 2250
2340 IF OB(N)=SP THEN ER=-1 : GOTO 2310
2350 IF BD$(X)<>"I"THEN 2270
2360 IF OB(N)=-1 THEN ER=-1 : GOTO 2310
2370 IF BD$(X)<>"N"THEN 2290
2380 IF (OB(N)<>SP AND OB(N)]<>-1) THEN
ER=-1 : GOTO 2310
2390 IF BD$(X)<>"S"THEN 2410
2400 R1$=BD$(X+1)+BD$(X+2):X=X+2:IF SP=V
AL(R1$) THEN ER=-1 : GOTO 2310
2410 IF BD$(X)<>"F" THEN 2450
2420 IF FL(VAL(BD$(X+1))) THEN 2440
2430 X=X+1:GOTO 2450
2440 ER=-1:X=X+1:GOTO 2310
2450 IF BD$(X)<>"G" THEN 2310
2460 IF NOT FL(VAL(BD$(X+1))) THEN 2480
2470 X=X+1:GOTO 2310
2480 ER=-1:X=X+1:GOTO 2310
2490 GOTO 2310
2500 IF NOT ER THEN PRINT"DAS GEHT IM MO
MENT NICHT.":GOTO 1080
3999 REM ---- ALLE BEDINGUNEGEN ERFUELLT
4000 PRINT"O.K."
4999 REM ------------ AKTIONEN AUSFUEHREN
5000 X=0:ER=0
5040 X=X+1
5050 IF X>AA THEN 1080
5060 IF AD$(X)<>"V" THEN 5080
```

```
5070 OB(N)=0:GOTO 5040
5080 IF AD$(X)<>"I" THEN 5100
5090 OB(N)=-1:GOTO 5040
5100 IF AD$(X)<>"N" THEN 5120
5110 GOSUB 5500:OB(PA)=SP:GOTO 5040
5120 IF AD$(X)<>"F" THEN 5140
5130 GOSUB 5600:FL(PA)=-1:GOTO 5040
5140 IF AD$(X)<>"L" THEN 5160
5150 GOSUB 5600:FL(PA)=0 :GOTO 5040
5160 IF AD$(X)<>"M" THEN 5180
5170 GOSUB 5500:PRINTMS$(PA) :GOTO 5040
5180 IF AD$(X)<>"E" THEN 5200
5190 GOTO 6000
5200 IF AD$(X)<>"D"THEN 5040
5210 GOTO 5700
5220 DU(SP,P1)=P2
5230 IF P1=1 THEN P3=2
5240 IF P1=2 THEN P3=1
5250 IF P1=3 THEN P3=4
5260 IF P1=4 THEN P3=3
5270 IF P1=5 THEN P3=6
5280 IF P1=6 THEN P3=5
5290 DU(P2,P3)=SP:GOTO5040
5500 R1$=AD$(X+1)+AD$(X+2):X=X+2:PA=VAL(
R1$):RETURN
5600 R1$=AD$(X+1):X=X+1:PA=VAL(R1$):RETU
RN
5700 P1=VAL(AD$(X+1)):P2=VAL(AD$(X+2)):X
=X+2:RETURN
6000 PRINT"■HERZLICHEN GLUECKWUNSCH !"
6010 PRINT"■■■DU HAST ES GESCHAFFT !"
6020 PRINT"■■■■■■■■":END

READY.
```

STICHWORTVERZEICHNIS

B

C

D

E

F

G

H

I

K

L

M

O

P

R



DATA BECKER'S NEUE BÜCHER UND PROGRAMME FÜR COMMODORE

Spickzettel ade.



Ein neues DATA BECKER BUCH, das den Einsatz des COMMODORE 64 in der Schule entscheidend mitprägen dürfte, wurde von Professor Voß geschrieben. Besonders für Schüler der Mittel- und Oberstufe geschrieben, enthält das Buch viele interessante Problemlösungs- und Lernprogramme, die besonders ausführlich und leicht verständlich beschrieben sind. Sie ermöglichen ein intensives und anregendes Lernen, unter anderem mit folgenden Themen: Satz des Pythagoras, quadratische Gleichungen, geometrische Reihen, Pendelbewegungen, mechanische Hebel, Molekülbildung, exponentielles Wachstum, Vokabeln lernen, unregelmäßige Verben, Zinseszinsrechnung. Ein kurzer Überblick über die Grundlagen der EDV, eine knappe Wiederholung der wichtigsten BASIC-Elemente und eine Einführung in die Grundzüge der Problemanalyse vervollständigen das Ganze. Mit diesem Buch machen die Hausaufgaben wieder Spaß!

DAS SCHULBUCH ZUM COMMODORE 64, 1984, über 300 Seiten, DM 49,–

Tempo!



MASCHINENSPRACHE FÜR FORTGESCHRITTENE ist bereits das zweite Buch von Lothar Englisch zum Thema Maschinenprogrammierung mit dem COMMODORE 64. Hier wird von der Problemanalyse bis zum Maschinensprachealgorithmus in die Grundlagen der professionellen Maschinensprache-programmierung eingeführt. In diesem Buch finden Sie unter anderem folgende Themen behandelt: Problemlösungen in Maschinensprache, Programmierung von Interruptroutinen, Interruptquellen beim COMMODORE 64, Interrupts durch CIA's und Videocontroller, Programmierung der Ein-Ausgabe-Bausteine, die CIA's des COMMODORE 64, Timer, Echtzeituhr, parallele und serielle Ein/Ausgabe, BASIC-Erweiterungen, Programmierung eigener BASIC-Befehle und -Funktionen, Möglichkeiten zur Einbindung ins Betriebssystem sowie viele weitere Tips & Tricks zur Maschinenprogrammierung. Dieses Buch sollte jeder haben, der wirklich intensiv mit der Maschinensprache des COMMODORE 64 arbeiten will.

MASCHINENSPRACHE FÜR FORTGESCHRITTENE, 1984, ca. 200 Seiten, DM 39,–

Macht Druck.



DAS GROSSE DRUCKERBUCH für Drucker-Anwender mit COMMODORE-Computern ist endlich da! Es enthält eine riesige Sammlung von Tips & Tricks, Programmlistings und Hardwareinformationen. Rolf Brückmann und Klaus Gerits beschäftigen sich mit Sekundäradressen, Anschluß einer Schreibmaschine am Userport, Druckerschnittstellen (Centronics, V 24, IEC-Bus), hochauflösender Grafik, Text- und Grafikhardcopy, Grafik mit Standardzeichensatz, formatierter Datenausgabe, Plakatschrift, Textverarbeitung und vieles mehr. Zusätzlich wird das Betriebssystem des MPS801 zerlegt, mit Prozessorbeschreibung (8035), Blockschaltbild und einem ausführlich kommentierten ROM-Listing. Thomas Wiens schrieb den Teil über die Programmierung des Plotters VC-1520: Handhabung des Plotters, Programmierung von Sonderzeichen, Funktionendarstellung, Kuchen und Säulendiagramme, Entwurf dreidimensionaler Gegenstände. Natürlich wieder viele interessante Listings. Unentbehrlich für jeden, der einen COMMODORE 64 oder VC-20 und einen Drucker besitzt.

DAS GROSSE DRUCKERBUCH, 1984, über 300 Seiten, DM 49,–

Tausendsassa.



Fast alles, was man mit dem COMMODORE 64 machen kann, ist in diesem Buch ausführlich beschrieben. Es ist nicht nur spannend zu lesen wie ein Roman, sondern enthält neben nützlichen Programmlistings vor allem viele, viele Anwendungsmöglichkeiten des C64. Dabei wurde besonderer Wert darauf gelegt, daß das Buch auch für Laien leicht verständlich ist. Eine Auswahl aus der Themenvielfalt: Gedichte vom Computer, Einladung zur Party, Diplomarbeit – professionell gestaltet, individuelle Werbebriefe, Autokosten im Griff, Baukostenberechnung, Taschenrechner, Rezeptkartei, Lagerliste, persönliches Gesundheitsarchiv, Diätplan elektronisch, intelligentes Wörterbuch, kleine Notenschule, CAD für Handarbeit, Routenoptimierung, Schaufensterwerbung, Strategiespiele. Teilweise sind Programmlistings fertig zum Eintippen enthalten, soweit sich die „Rezepte" auf 1–2 Seiten realisieren ließen. Wenn Sie bisher nicht immer wußten, was Sie mit Ihrem 64er alles anfangen sollten, nach dem Lesen des IDEENBUCHES wissen Sie's bestimmt!

DAS IDEENBUCH ZUM COMMODORE 64, 1984, über 200 Seiten, DM 29,–

Prof. 64.



Ein faszinierendes Buch, um in die Welt der Wissenschaft einzusteigen, hat Rainer Severin geschrieben. Zunächst werden Variablentypen, Rechengenauigkeit und nützliche POKE-Adressen des COMMODORE 64 bezüglich den Anforderungen wissenschaftlicher Probleme analysiert. Verschiedene Sortieralgorithmen wie Bubble, Quick und Shell-Sort werden miteinander verglichen. Die Programmbeispiele aus der Mathematik nehmen dabei eine zentrale Stelle im Buch ein: Nullstellen nach Newton, numerische Ableitung mit dem Differenzenquotienten, lineare und nichtlineare Regression, Chi-Quadrat-Verteilung und Anpassungstest, Fourieranalyse und -synthese, Skalar-, Vektor- und Spatprodukt, ein Programmpaket zur Matrizenrechnung für Inversion, Eigenwerte und vieles weitere mehr. Programme aus der Chemie (Periodensystem), Physik, Biologie (Schadstoffe in Gewässern – Erfassung der Meßwerte), Astronomie (Planetenpositionen) und Technik (Berechnung komplexer Netzwerke, Platinenlayout am Bildschirm) und viele weitere Softwarelistings zeigen die riesigen Möglichkeiten auf, die der Computer in Wissenschaft und Technik hat.

COMMODORE 64 FÜR TECHNIK UND WISSENSCHAFT, 1984, über 200 Seiten, DM 49,–
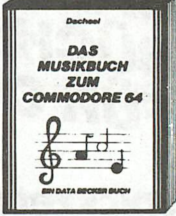
Grundkurs.



Das neue BASIC-Trainingsbuch zum C-64 ist eine ausführliche, didaktisch gut geschriebene Einführung in das CBM BASIC V2. Alle Befehle werden ausführlich erläutert. Dieses Buch geht aber über eine reine Befehlsbeschreibung hinaus, es wird eine fundierte Einführung in die Programmierung gegeben. Von der Problemanalyse bis zum fertigen Algorithmus lernt man das Entwerfen eines Programmes und den Entwurf von Datenflußplänen. ASCII-Code und verschiedene Zahlensysteme wie hexadezimal, binär und dezimal sind nach der Lektüre des Buches keine Fremdworte mehr. Die Programmierung von Schleifen, Sprüngen, bedingten Sprüngen lernt man leicht durch „learning by doing". So enthält das Trainingsbuch viele Aufgaben, Übungen und unzählige Beispiele. Den Schluß des Buches bildet eine Einführung ins professionelle Programmieren, in der es um mehrdimensionale Felder, Menuesteuerung und Unterprogrammtechnik geht. Endlich ein Buch, das Ihnen wirklich hilft, solide und sicher BASIC zu lernen.

BASIC TRAININGSBUCH ZUM COMMODORE 64, 1984, ca. 250 Seiten, DM 39,–

Sang und Klang!



Der COMMODORE 64 ist ein Musikgenie. DAS MUSIKBUCH hilft Ihnen, die riesigen Klangmöglichkeiten des C 64 zu nutzen. Die Themenbreite reicht von einer Einführung in die Computermusik über die Erklärung der Hardwaregrundlagen des COMMODORE 64 und die Programmierung in BASIC bis hin zur fortgeschrittenen Musikprogrammierung in Maschinensprache. Einiges aus dem Inhalt: Soundregister des COMMODORE 64, Gate-Signal, Programmierung der "ADSR"-Werte, Synchronisation und Ring-Modulation, Counterprinzip, lineare und nichtlineare Musikprogrammierung, Frequenzmodulation, Interrupts in der Musikprogrammierung und vieles mehr. Zahlreiche Beispielprogramme, komplette Songs und nützliche Routinen ergänzen den Text. Geschrieben wurde das Buch von Thomas Dachsel, dem Autor der weltbekannten Musikprogramme Synthimat und Synthesound. Erschließen Sie sich die Welt des Sounds und der Computermusik mit dem Musikbuch zum C-64!

DAS MUSIKBUCH ZUM COMMODORE 64, über 200 Seiten, DM 39,–

Nützlich.



Das Trainingsbuch zu MULTIPLAN bietet eine gute Einführung in die Grundlagen der Tabellenkalkulation. Dabei wird großer Wert auf ein möglichst schnelles Einarbeiten in die wichtigsten Befehle gelegt, so daß man bald sicher mit MULTIPLAN arbeiten kann, ob nun auf dem COMMODORE 64 oder einem anderen Rechner. Am Ende wird man in der Lage sein, den umfangreichen Befehlssatz von MULTIPLAN auch kommerziell zu nutzen. Übungen am Ende jedes Kapitels sorgen dafür, daß man das Gelernte lange behält. Grundlage des Buches sind viele Seminare, die der Autor zu MULTIPLAN konzipiert und erfolgreich durchgeführt hat.

DAS TRAININGSBUCH ZU MULTIPLAN, 1984, ca. 250 Seiten, DM 49,–

Für Tüftler.



Ein hochinteressantes Buch für Hobbyelektroniker hat Rolf Brückmann vorgelegt. Er ist ein engagierter Techniker, für den der Computer Hobby und Beruf zur gleichen Zeit ist. Vor allem aber kennt er den C-64 in- und auswendig. So werden einführend die Schnittstellen des COMMODORE 64 detailliert beschrieben und kurz die Funktionsweise der CIAs 6526 erläutert. Hauptteil des Buches sind die Beschreibungen der vielfältigen Einsatzmöglichkeiten des COMMODORE 64. Die vielen Schaltungen, von Rolf Brückmann alle selbst

entwickelt, sind jeweils umfangreich dokumentiert und leichtverständlich erklärt. Die Reihe der hier ausführlich behandelten Anwendungen mit dem COMMODORE 64 ist äußerst umfangreich: Motorsteuerung, Stoppuhr mit Lichtschranke, Lichtorgel, A/D-Wandler, Spannungsmessung, Temperaturmessung und vieles mehr. Dazu kommen noch eine Reihe kompletter Schaltungen zum Selberbauen, wie ein EPROM Programmiergerät für den C-64, eine EPROM-Karte, ein Frequenzzähler und Sprachein/ausgabe (!). Zusätzlich sind jeweils Schaltplan, Softwarelisting und zu einigen Schaltungen sogar zusätzlich Platinenlayouts vorhanden.

DER COMMODORE 64 UND DER REST DER WELT, 1984, ca. 220 Seiten, DM 49,–

Computerkünstler.

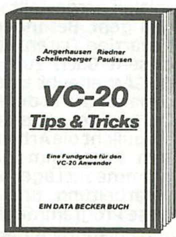
Das Grafikbuch zum COMMODORE 64 Buch aus der Bestseller-Serie von DATA BECKER stammt aus der Feder von Axel Plenge. Es geht weit über die reine Hardware-Beschreibung der Grafikeigenschaften des C-64 hinaus. Der Inhalt reicht von den Grundlagen der Grafikprogrammierung bis zum Computer Aided Design. Es ist ein Buch für alle, die mit ihrem C-64 kreativ tätig sein wollen. Themen sind z. B.: Zeichensatzprogrammierung, bewegte Sprites, High-Resolution, Multicolor-Grafik, Lightpenanwendungen, Betriebsarten des VIC, Verschieben der Bildschirmspeicher, IRQ-Handhabung, 3-Dimensionale Grafik, Projektionen, Kurven-, Balken- und Kuchendiagramme, Laufschriften, Animation, bewegte Bilder. Viele Programmlistings und Beispiele sind selbstverständlich. Das COMMODORE-BASIC V2 unterstützt die herausragenden Grafikeigenschaften des C-64 bekanntlich kaum. Hier helfen die vielen Beispielprogramme in diesem Buch weiter, die die faszinierende Welt der Computergrafik jedermann zugänglich machen. Kompetent ist der Autor dazu wie kaum ein anderer, schließlich hat er das äußerst leistungsfähige Programm SUPERGRAFIK geschrieben.



DAS GRAFIKBUCH ZUM COMMODORE 64, 1984, 295 Seiten, DM 39,–

Vielfalt.

Auf dem neuesten Stand ist VC-20 TIPS & TRICKS von Dirk Paulissen gebracht worden, der über hundert Seiten hinzufügte. Bisher schon enthalten waren Informationen über Speicheraufbau des VC-20 und die Erweiterungsmöglichkeiten, ein Grafikkapitel über programmierbare Zeichen, Laufschrift und die Supererweiterung. Stark erweitert wurde der Abschnitt über POKEs und andere nützliche Routinen. Ob es um die Programmierung der Funktionstasten, Programme die sich selber starten, „Maus"-Simulation mit dem Joystick oder die Änderung von Speicherbereichen geht, man ist immer wieder über die Fülle der Möglichkeiten erstaunt. Der Clou dieses Buches sind aber die vielen Programmlistings. Die BASIC-Erweiterungen allein stellen schon ein erstklassiges Toolkit dar: APPEND (Anhängen von Programmen, AUTO (automatische Zeilennummerierung), BASIC-Befehle auf Tastendruck, PRINT POSITION, UNNEW, Strings größer als 88 Zeichen einlesen und vieles mehr. Die Bandbreite reicht von Spielen wie Goldgräber oder Starshooter bis zu nützlichen Programmen wie Cassetteninhaltsverzeichnis und -katalog mit automatischem Suchen nach Dateien und einem Terminkalender. Für den VC-20 Anwender ist dieser 324 Seiten-Wälzer eine wahre Fundgrube, in der es immer etwas neues zu entdecken gibt.



VC-20 TIPS & TRICKS, 3. erweiterte und überarbeitete Auflage, 1984, 324 Seiten, DM 49,–

Interessant.

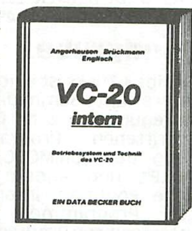
Einen guten Einstieg in PASCAL bietet dieses Trainingsbuch. Es gibt eine leichtverständliche Einführung, sowohl in UCSD-PASCAL wie auch in PASCAL64, wobei allerdings EDV-und BASIC-Grundkenntnisse vorausgesetzt werden. Der Autor, Ottmar Korbmacher, ist Student der Mathematik. Ihm gelingt es, in einem sprachlich aufgelockerten Stil mit vielen interessanten Beispielprogrammen, dem Leser Programmstrukturen, Ein/Ausgabe, Arithmetik und Funktionen, Prozeduren und Rekursionen, Sets, Files und Records näherzubringen. Die Übungsaufgaben am Ende jeden Kapitels helfen dabei, das Gelernte zu vertiefen. Ein Anhang mit allen PASCAL-Schlüsselworten, der ansich schon ein umfangreiches Lexikon darstellt, macht das Buch für jeden PASCAL-Anwender interessant.



DAS TRAININGSBUCH ZU PASCAL, 1984, ca. 250 Seiten, DM 39,–

Bewährt.

Die bereits dritte Auflage von VC-20 INTERN ist wieder erheblich erweitert worden. Das Buch beschäftigt sich ausführlich mit der Technik und dem Betriebssystem des VC-20. Dazu gehört natürlich zuerst einmal ein ausführlich dokumentiertes ROM-Listing. Dazu gehört auch die Belegung der Zeropage, dem wichtigsten Speicherbereich für den 6502-Prozessor, eine übersichtliche Auflistung der Adressen aller Betriebssystemroutinen, ihrer Bedeutung und ihrer Übergabeparameter. Dies ermöglicht dem Programmierer endlich, den VC-20 von Maschinensprache aus sinnvoll einzusetzen. Denn warum Routinen, die bereits vorhanden sind, noch einmal schreiben? Weiterer Inhalt: Einführung in die Maschinensprache – Maschinensprachemonitor, Assembler, Disassembler – Verbindung von Maschinensprache- und BASIC-Programmen – Beschreibung der wichtigen IC's des VC-20 – Blockschaltbild – drei Original COMMODORE-Schaltpläne. Das Buch braucht jeder der sich intensiv mit der Maschinenspracheprogrammierung des VC-20 auseinandersetzen möchte.



VC-20 INTERN, 3. Auflage, 1984, ca. 230 Seiten, DM 49,–

Starthilfe!



Das sollte Ihr erstes Buch zum COMMODORE 64 sein: 64 FÜR EINSTEIGER ist eine sehr leicht verständliche Einführung in Handhabung, Einsatz, Ausbaumöglichkeiten und Programmierung des COMMODORE 64, die keinerlei Vorkenntnisse voraussetzt. Sie reicht vom Anschluß des Geräts über die Erklärung der einzelnen Tasten und Funktionen sowie die Peripheriegeräte und ihre Bedienung bis zum ersten Befehl. Schritt für Schritt führt das Buch Sie in die Programmiersprache BASIC ein, wobei Sie nach und nach eine komplette Adressenverwaltung erstellen, die Sie anschließend nutzen können. Zahlreiche Abbildungen und Bildschirmfotos ergänzen den Text. Viele Anwendungsbeispiele geben nützliche Anregungen zum sinnvollen Einsatz des COMMODORE 64. Das Buch ist sowohl als Einführung als auch als Orientierung vor dem 64er Kauf gut geeignet.

64 FÜR EINSTEIGER, 1984, ca. 200 Seiten, DM 29,–

Von A bis Z.

So etwas haben Sie gesucht: Umfassendes Nachschlagewerk zum COMMODORE 64 und seiner Programmierung. Allgemeines Computerlexikon mit Fachwissen von A–Z und Fachwörterbuch mit Übersetzungen wichtiger englischer Fachbegriffe – das DATA BECKER LEXIKON ZUM COMMODORE 64 stellt praktisch drei Bücher in einem dar. Es enthält eine unglaubliche Vielfalt an Informationen und dient so zugleich als kompetentes Nachschlagewerk und als unentbehrliches Arbeitsmittel. Viele Abbildungen und Beispiele ergänzen den Text. Ein Muß für jeden COMMODORE 64 Anwender!



DAS DATA BECKER LEXIKON ZUM COMMODORE 64, 1984, 354 Seiten, DM 49,–

Fundgrube.



64 Tips & Tricks ist eine hochinteressante Sammlung von Anregungen zur fortgeschrittenen Programmierung des COMMODORE 64, POKE's und andere nützliche Routinen, interessanten Programmen sowie interessanten Programmiertips & -tricks. Aus dem Inhalt: 3D-Graphik in BASIC – Farbige Balkengraphik – Definition eines eigenen Zeichensatzes – Tastaturbelegung und ihre Änderung – Dateneingabe mit Komfort – Simulation der Maus mit einem Joystick – BASIC für Fortgeschrittene – C-64 spricht deutsch – CP/M auf dem COMMODORE 64 – Druckeranschluß über den USER-Port – Datenübertragung von und zu anderen Rechnern – Expansion-Port – Synthesizer in Stereo – Retten einer nicht ordnungsgemäß geschlossenen Datei – Erzeugen einer BASIC-Zeile in BASIC – Kassettenpuffer als Datenspeicher – Sortieren von Stringfelder – Multitasking auf dem COMMODORE 64 – POKE's und die Zeropage – GOTO, GOSUB und RESTORE mit berechneten Zeilennummern, INSTR und STRING-Funktion – Repeat-Funktion für alle Tasten – und vieles andere mehr. Alle Maschinenprogramme mit BASIC-Ladeprogrammen. 64 Tips & Tricks ist eine echte Fundgrube für jeden COMMODORE 64 Anwender. Schon über 65000mal verkauft!

64 TIPS & TRICKS, 1984, über 300 Seiten, DM 49,–

Know-how!



350 Seiten dick ist die 4. erweiterte und überarbeitete Auflage von 64 INTERN geworden. Das bereits über 65000mal verkaufte Standardwerk bietet jetzt noch mehr Informationen. Hinzugekommen ist ein Kapitel über den IEC-Bus und viele, viele Ergänzungen, die sich im Laufe der Zeit angesammelt haben. Ebenfalls überarbeitet und noch ausführlicher ist jetzt die Dokumentation des ROM-Listings. Weitere Themen: genaue Beschreibung des Sound- und Video-Controllers mit vielen Hinweisen zur Programmierung von Sound und Grafik, der Ein/Ausgabesteuerung (CIAs), BASIC-Erweiterungen (RENEW, HARDCOPY, PRINTUSING), Hinweise zur Maschinenprogrammierung wie Nutzung der E/A-Routinen des Betriebssystems, Programmierung der Schnittstelle RS 232, ein Vergleich VC20 – C-64 – CBM zur Umsetzung von Programmen. Dies und viele weitere Informationen machen das umfangreiche Werk zu einem unentbehrlichen Arbeitsmittel für jeden, der sich ernsthaft mit Betriebssystem und Technik des C-64 auseinandersetzen will. Zum professionellen Gehalt des Buches tragen auch zwei Original-COMMODORE-Schaltpläne zum Ausklappen und zahlreiche ausführlich beschriebene und dokumentierte Fotos, Schaltbilder und Blockdiagramme bei.

64 INTERN, 4. überarbeitete und erweiterte Auflage, 1984, ca. 350 Seiten, DM 69,–

Erfolgreich.



64 für Profis zeigt, wie man erfolgreich Anwendungsprobleme in BASIC löst und verrät die Erfolgsgeheimnisse der Programmierprofis. Vom Programmentwurf über Menüsteuerung, Maskenaufbau, Parametrisierung, Datenzugriff und Druckausgabe bis hin zur guten Dokumentation wird anschaulich mit vielen Beispielen dargestellt wie Profi-Programmierung vor sich geht. Besonders stolz sind wir auf die völlig neuartige Datenzugriffsmethode QUISAM, die in diesem Buch zum ersten Mal vorgestellt wird. QUISAM erlaubt eine beliebige Datensatzlänge, die dynamisch mit der Eingabe der Daten wächst. Eine lauffertige Literaturstellenverwaltung veranschaulicht die Arbeitsweise von QUISAM. Neben diesem Programm finden Sie noch weitere Programme zur Lager- und Adressenverwaltung, Textverarbeitung und einen Reportgenerator. Alle diese Programme sind mit Variablenliste versehen und ausführlich beschrieben. Damit sind diese für Ihre Erweiterungen offen und können von Ihnen an Ihre persönlichen Bedürfnisse angepaßt werden. Steigen Sie in die Welt der Programmierprofis ein.

64 FÜR PROFIS, 2. Auflage, 1984, ca. 300 Seiten, DM 49,–

Rundum gut!



Endlich ein Buch, das Ihnen ausführlich und verständlich die Arbeit mit der Floppy VC-1541 erklärt. Das große Floppybuch ist für Anfänger, Fortgeschrittene und Profis gleichermaßen. interessant. Sein Inhalt reicht von der Programmspeicherung bis zum DOS-Zugriff, von der sequentiellen Datenspeicherung bis zum Direktzugriff, von der technischen Beschreibung bis zum ausführlich dokumentierten DOS-Listing, von den Systembefehlen bis zur detaillierten Beschreibung der Programme auf der Test-Demo-Diskette. Exakt beschriebene Beispiel- und Hilfsprogramme ergänzen dieses neue Superbuch. Aus dem Inhalt: Speichern von Programmen – Floppy-Systembefehle – Sequentielle Datenspeicherung – relative Datenspeicherung – Fehlermeldungen und ihre Ursachen – Direktzugriff – DOS-Listing der VC-1541 – BASIC-Erweiterungen und Programme – Overlaytechnik – Diskmonitor – IEC-Bus und serieller Bus – Vergleich mit den großen CBM-Floppies. Ein Muß für jeden Floppy-Anwender! Bereits über 45.000mal verkauft.

DAS GROSSE FLOPPY-BUCH, 2. überarbeitete Auflage, 1984, ca. 320 Seiten, DM 49,–

Füttern erwünscht!
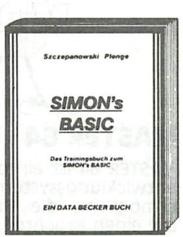


Diese beliebte umfangreiche Programmsammlung hat es in sich. Über 50 Spitzenprogramme für den COMMODORE 64 aus den unterschiedlichsten Bereichen, von attraktiven Superspielen (Senso, Pengo, Master Mind, Seeschlacht, Poisson Square, Memory) über Grafik- und Soundprogramme (Fourier 64, Akustograph, Funktionsplotter) und mathematische Programme (Kurvendiskussion, Dreieck) sowie Utilities (SORT, RENUMBER, DISK INIT, MENUE) bis hin zu kompletten Anwendungsprogrammen wie „Videothek", „File Manager" und einer komfortablen Haushaltsbuchführung, in der fast professionell gebucht wird. Der Hit zu jedem Programm sind aktuelle Programmiertips und Tricks der einzelnen Autoren zum Selbermachen. Also nicht nur abtippen, sondern auch dabei lernen und wichtige Anregungen für die eigene Programmierung sammeln.

DATA BECKER's GROSSE 64er PROGRAMMSAMMLUNG, 1984, 250 Seiten, DM 49,–

Bestseller aus bester Hand

BASIC-PLUS.



SIMON's BASIC ist ein Hit – wenn man es richtig nutzen kann. Auf über 300 Seiten erklärt Ihnen das DATA BECKER Trainingsbuch detailliert den Umgang mit den über 100 Befehlen des SIMON's BASIC. Alle Befehle werden ausführlich dargestellt, auch die, die nicht im Handbuch stehen! Natürlich zeigen wir auch die Macken des SIMON's BASIC und geben wichtige Hinweise wie man diese umgeht. Natürlich enthält das Buch viele Beispielprogramme und viele interessante Programmiertricks. Weiterer Inhalt: Einführung in das CBM-BASIC 2.0 – Programmierhilfen – Fehlerbehandlung – Programmschutz – Programmstruktur – Variablen – Zahlenbehandlung – Eingabekontrolle – Ein/Ausgabe Peripheriebefehle – Graphik – Zeichensatzerstellung – Sprites – Musik – SIMON's BASIC und die Verträglichkeit mit anderen Erweiterungen und Programmen. Dazu ein umfangreicher Anhang. Nach jedem Kapitel finden Sie Testaufgaben zum optimalen Selbststudium und zur Lernerfolgskontrolle.

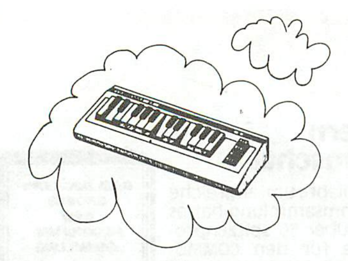
DAS TRAININGSBUCH ZUM SIMON's BASIC, 2. überarbeitete Auflage, 1984, ca. 380 Seiten, DM 49,–

Schrittmacher.



Eine leicht verständliche Einführung in die Maschinensprachеprogrammierung für alle, denen das C-64 BASIC nicht mehr ausreicht. Sie lernen Aufbau und Arbeitsweise des 6510-Mikroprozessors kennen und anwenden. Dabei werden die Analogien zu BASIC Ihnen beim Verständnis helfen. Ein weiteres Kapitel beschäftigt sich mit der Eingabe von Maschinenprogrammen. Dort erfahren Sie auch alles über Monitor-Programme sowie über Assembler. Zum einfachen und komfortablen Erstellen Ihrer eigenen Maschinensprache enthält das Buch einen kompletten ASSEMBLER, damit Sie gleich von Anfang an komfortabel und effektiv programmieren können. Weiterhin finden Sie dort einen DISASSEMBLER, mit dem Sie sich Ihre Maschinenprogramme oder die Routinen des BASIC-Interpreters und des BASIC-Betriebssystems ansehen können. Ein besonderer Clou ist ein in BASIC geschriebener Einzelschrittsimulator, mit dem Sie Ihre Programme schrittweise ausführen können. Dabei werden Sie nach jedem Schritt über Registerinhalte und Flags informiert und können den logischen Ablauf Ihres Programmes verfolgen. Eine unschätzbare Hilfe, besonders für den Anfänger. Als Beispielprogramm finden Sie ausführlich beschriebene Routinen zur Grafikprogrammierung und für BASIC-Erweiterungen. Natürlich sind alle Beispiele und Programme auf den C-64 zugeschnitten.

DAS MASCHINENSPRACHEBUCH ZUM COMMODORE 64, ca. 200 Seiten, DM 39,–



SYNTHIMAT

SYNTHIMAT verwandelt Ihren COMMODORE 64 in einen professionellen, polyphonen, dreistimmigen Synthesizer, der in seinen unglaublich vielen Möglichkeiten großen Systemen kaum nachsteht.

SYNTHIMAT in Stichworten:
drei Oszillatoren (VCOs) mit 7 Fußlagen und 8 Wellenformen – drei Hüllkurvengeneratoren (ADSRs) – ein Filter (VCF) mit 8 Betriebsarten und Resonanzregulierung – VCF mit Eingang für externe Signalquelle – ein Verstärker (VCA) – Ringmodulation mit allen drei VCOs – 8 softwaremäßig realisierte Oszillatoren (LFOs) – kräftiger Klang durch polyphones Spielen – zwei Manuale (Solo und Begleitung) – speichern von bis zu 256 Klangregistern – schneller Registerwechsel – speichern von 9 Registerdateien auf Diskette – „Bandaufnahme" auf Diskette durch direktes Spielen – keine lästige Noteneingabe – speichern von bis zu 9 „Bandaufnahmen" je Diskette – integrierte 24 Stunden-Echtzeituhr – einstellbares PITCH-BENDING – farblich gekennzeichnete, übersichtlich angeordnete Module – umfangreiches Handbuch – läuft mit einem Diskettenlaufwerk – Diskettenprogramm.

DM 99,–



STRUKTO 64

STRUKTO 64 ist eine fantastische neue Programmiersprache für strukturiertes Programmieren mit dem C-64 und für alle Programmierer geeignet, die den C-64 als Allround-Computer einsetzen und auf einfache Weise anspruchsvolle Programme erstellen wollen.

STRUKTO 64 in Stichworten:
Interpretersprache, die die Vorzüge von BASIC und PASCAL vereint – strukturiertes Programmieren – übersichtliche Programme – leichte Erlernbarkeit – einfache Bedienung – eingebautes Toolkit erleichtert das Eingeben und Verbessern von Programmen – leichteres Arbeiten mit der Floppy – Sprite-Editor ermöglicht das Einlesen der Sprite-Formen direkt vom Bildschirm – Graphikbedienung wird mit gut durchdachten Befehlen unterstützt – Abspielen von Musik ist unabhängig vom Programmablauf möglich – ca. 80 neue Befehle – lieferbar als Diskettenprogramm – ausführliches deutsches Handbuch.

DM 99,–



Für viele ein Traum, für die meisten bisher zu teuer: die Rede ist von einer echten Datenbank für den 64er. SUPERBASE 64 füllt eine Lücke. Nicht allein die Kapazität, die verwaltet werden kann, bewegt sich in professionellen Regionen, die ausgeprägten Fähigkeiten des SUPERBASE 64 im Rechnen und Kalkulieren lassen dieses Paket beinahe als Rund-Um-Software erscheinen.

SUPERBASE 64 in Stichworten:
maximale Datensatzlänge 1108 Zeichen, verteilt auf bis zu 4 Bildschirmseiten – bis zu 127 Felder pro Datensatz, wobei Textfelder bis zu 255 Zeichen lang sein können – insgesamt 15 Einzeldateien können zu einer SUPERBASE-Datenbank verknüpft werden – Speicherkapazität nur durch Diskette begrenzt – umfangreiche Auswertungsmöglichkeiten und komfortabler Report-Generator – Kalkulationsmöglichkeiten und Rechnen – Import- (Einlesen von externen Daten) und Export- (Ausgabe von SUPERBASE Dataien als sequentielle Datei) Funktionen ermöglichen Datenaustausch mit anderen Programmen – durch leistungsfähige, eigene Datenbanksprache auch als kompletter Anwendungsgenerator verwendbar.

DM 398,–



MASTER 64

MASTER 64 ist ein professionelles Programmentwicklungssystem für den C-64, das es Ihnen ermöglicht, die Programmentwicklungszeit auf einen Bruchteil der sonst üblichen Zeit zu reduzieren. MASTER 64 bietet einen Programmkomfort, den Sie nutzen sollten.

MASTER 64 in Stichworten:
70 zusätzliche Befehle – Bildschirmmaskengenerator – definieren von Bildschirmzonen – Eingabe aus Zonen – formatierte Ausgabe – Abspeicherung von Bildschirminhalten – Arbeiten mit mehreren Bildschirmmasken – ISAM Dateiverwaltung, in der Datensätze über einen Zugriffschlüssel angesprochen werden können – Datensätze bis zu 254 Zeichen – Schlüssellänge bis zu 30 Zeichen – Dateigröße nur von Diskettenkapazität abhängig – Zugriff über Schlüssel und Auswahlmasken – Bildschirm- und Druckmaskengenerator – Erstellung beliebiger Formulare und Ausgabemasken – BASIC-Erweiterungen – Toolkitfunktionen – Mehrfachgenaue Arithmetik (Rechnen mit 22 Stellen Genauigkeit).

DM 198,–



DAS STEHT DRIN:

Ein faszinierender Führer in die phantastische Welt der Abenteuer-Spiele. Hier wird gezeigt wie Adventures funktionieren, wie man sie erfolgreich spielt und wie man eigene Adventures auf dem COMMODORE 64 programmiert. Der Clou des Buches ist neben vielen fertigen Adventures zum Abtippen ein kompletter ADVENTUREGENERATOR, mit dem das Selberprogrammieren packender Adventures zum Kinderspiel wird.

UND GESCHRIEBEN HAT DIESES BUCH:

Jörg Walkowiak ist Informatik-Student und erfolgreicher Autor professioneller Adventures.

ISBN 3-89011-043-6